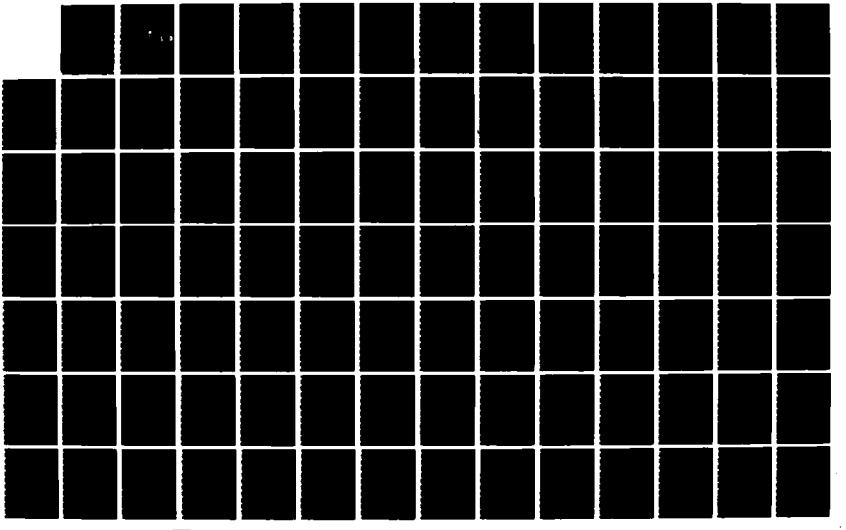
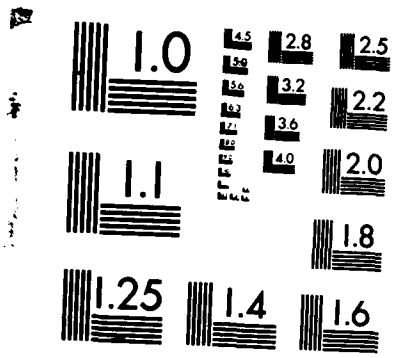
AD-A173 612 DEVELOPMENT OF A SHIPBOARD DAMAGE CONTROL FIRE TEAM LEADER INTELLIGENT CO (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S G WEINGART 20 JUN 86 1/2

UNCLASSIFIED F/G 13/12 NL

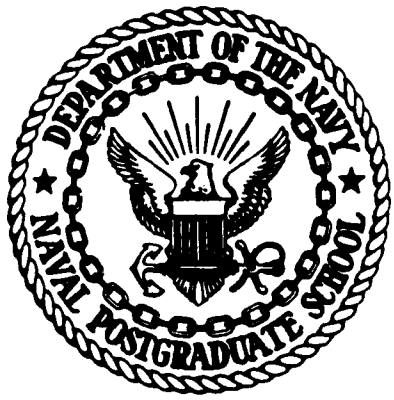




ICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A173 612

NAVAL POSTGRADUATE SCHOOL
Monterey, California



DTIC
ELECTE
NOV 05 1986
S D
E

THESIS

DEVELOPMENT OF A SHIPBOARD DAMAGE CONTROL FIRE TEAM LEADER INTELLIGENT COMPUTER AIDED INSTRUCTIONAL TUTORING SYSTEM

by

Stephen G. Weingart

June 1986

Thesis Advisor: Neil C. Rowe

Approved for public release; distribution is unlimited.

DTIC FILE COPY

86 11 6 007

SECURITY CLASSIFICATION OF THIS PAGE AD-A173612

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 52	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING / SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification) UNCLASSIFIED Development of a Shipboard Damage Control Fire Team Leader Intelligent Computer Aided Instructional Tutoring System

12 PERSONAL AUTHOR(S)
Stephen G. Weingart

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Masters Thesis	FROM _____ TO _____	1986 June 20	151

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Intelligent Computer-Aided Instructional, tutor, Fire Team Leader, Damage Control, Fire, Computer game, expert, behavioral difference

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

We discuss the design and implementation of FIRE, an Intelligent Computer-Aided Instructional (ICAI) tutoring system. It tutors Fire Team Leaders on U.S. Naval ships in fire control, including dressing personnel, setting fire boundaries, extinguishing fires, performing gas tests, debriefing personnel, and recovering from personnel injuries and broken equipment. A computer game environment challenges the user with a random fire scenario at the user's experience level. Using a multi-level tree of expert recovery actions, Fire can correctly choose the next best recovery action in any random fire scenario. Every incorrect student answer causes a formulation of a hypothesis concerning the cause of the behavioral difference between the student and an expert. This hypothesis guides selection of one of six possible tutoring strategies. The user can also perform a text-book-type lookup of correct action.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Neil C. Rowe	22b TELEPHONE (Include Area Code) 408-646-2462 / 22c OFFICE SYMBOL 52Rp

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

Development of a Shipboard Damage Control Fire Team Leader Intelligent Computer Aided Instructional Tutoring System

by

Stephen G. Weingart
Lieutenant, United States Navy
B.S., United States Naval Academy, 1977

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1986

Author: Stephen G. Weingart
Stephen G. Weingart

Approved by: Neil C. Rowe
Neil C. Rowe, Thesis Advisor

Michael J. Zyda, Second Reader

Vincent Y. Lum, Chairman,
Department of Computer Science

Kneale T. Marshall
Kneale T. Marshall II,
Dean of Information and Policy Sciences

ABSTRACT

This thesis

We discusses the design and implementation of FIRE, an Intelligent Computer-Aided Instructional (ICAI) tutoring system. It tutors Fire Team Leaders on U.S. Naval ships in fire control, including dressing personnel, setting fire boundaries, extinguishing fires, performing gas tests, debriefing personnel, and recovering from personnel injuries and broken equipment. A computer game environment challenges the user with a random fire scenario at the user's experience level. Using a multi-level tree of expert recovery actions, Fire can correctly choose the next best recovery action in any random fire scenario. Every incorrect student answer causes a formulation of a hypothesis concerning the cause of the behavioral difference between the student and an expert. This hypothesis guides selection of one of six possible tutoring strategies. The user can also perform a text-book-type lookup of correct action information.

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

QUALITY INSPECTED 4

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION 9

II. BACKGROUND 14

A. SHIPBOARD DAMAGE CONTROL ORGANIZATION FOR FIRE FIGHTING 14

B. SHIPBOARD DAMAGE CONTROL TRAINING PROGRAM . . 14

C. ALTERNATIVE TRAINING METHOD FOR INPORT FIRE TEAMS 17

1. ICAI Inport Fire Team Tutoring Systems . . 17

2. Analysis of the Role of a Fire Team Leader 17

III. FIRE: A FIRE TEAM LEADER LEARNING CENTER 21

A. FIRE SYSTEM CONFIGURATION AND DESCRIPTION . . 21

B. FIRE SYSTEM DETAILED DESIGN AND DESCRIPTION 24

1. Action Tree Model 24

2. Game Environment 35

3. Student/Expert Choice-Comparer 50

4. Student Model 50

5. Tutoring Module 54

C. SPECIAL SYSTEM MODES OF OPERATION 67

1. Individual Equipment Operations and Basic Damage Control Actions 67

2. Individual Complex Damage Control Actions 68

3. Complex Fire Casualty 69

4. Previous Fire Casualty Operation Training 70

IV. USING THE FIRE SYSTEM ON US NAVY SHIPS 71

A. MEMORY REQUIREMENTS 71

B. PROGRAM LANGUAGE REQUIREMENT 71

C. EXPANDING THE FIRE SYSTEM FOR DIFFERENT KNOWLEDGE DOMAINS 71

V. SUMMARY 73
A. DISCUSSION OF THE RESEARCH QUESTIONS 73
B. ICAI TUTORING SYSTEM LIMITATIONS 74
1. Realistic Simulation 74
2. Action Tree Model Accuracy 75
3. Knowledge Base Structure 76
4. Careless Errors 78
C. BENEFITS OF RESEARCH 79

APPENDIX A: DEMONSTRATION OF COMPLEX FIRE SIMULATION . . 80

APPENDIX B: EXPERT LEVEL DEMONSTRATION OF COMPLEX FIRE SIMULATION 99

APPENDIX C: DEMONSTRATION OF FIRE'S SECONDARY MODES OF OPERATION 108

APPENDIX D: FIRE'S MAJOR SUPPORTING PROGRAM FILES . . 119

LIST OF REFERENCES 147

BIBLIOGRAPHY 149

INITIAL DISTRIBUTION LIST 150

LIST OF FIGURES

2.1 Organization of a Large Fire Fighting Team 19

3.1 Fire Team Leader Learning Center System Configuration Diagram 22

3.2 Action Tree Model "treelike" Structure 26

3.3 Breakdown of an Expandable Action Node 27

3.4 Expandable Node Action File Framework 29

3.5 Breakdown of an Expandable Action Node to Non-Expandable Action Nodes 30

3.6 Breakdown of an Expandable Action Node to Test Procedure Action Nodes 31

3.7 Order Dependent Description Test Procedure Action File Framework 32

3.8 Informational Description Test Procedure Node Action File Framework 32

3.9 Partial Action Tree Model 36

3.10 Casualty Tree Model 38

3.11 Node Action Scene Report Predicate 40

3.12 Initial Seeding of the Random Task Generator Using the Time of Day 41

3.13 Display of Task Executor Directing User to Perform a Particular Task 43

3.14 Display of an Order Dependent Test Procedure Action Node 45

3.15 Display of an Informational Test Procedure Action Node 46

3.16 Examples of Casualty Scene Reports 46

3.17 User Performance Report 49

3.18 Help Facility Introduction 50

3.19 Student's Knowledge Tree Model 52

3.20 Wrong Task Tutoring Strategy 57

3.21 Weak Casualty Knowledge Tutoring Strategy 59

3.22 Forgetting Completed Action Tutoring Strategy . . . 60

3.23 Unsure Action Knowledge Tutoring Strategy 61

3.24 Transposition Error Tutoring Strategy 63
3.25 Examples of Bad Consequence Explanation Predicates 65
3.26 Incomplete Action Knowledge Tutoring Strategy . . . 66
3.27 Action Node Breakdown Feature Display One 66
3.28 Action Node Breakdown Feature Display Two 67
3.29 Action Node Breakdown Feature Display Three 68
3.30 Post Tutoring Module Trip Display 69

I. INTRODUCTION

The implementation of an effective damage control organization on US Naval Ships is the ultimate weapon to ensure the ship's survivability. Through realistic training for shipboard fire fighting groups, and using equipment that is maintained in an efficient operating condition, fire teams are able to successfully fight the most dangerous fires, thereby removing one of the greatest threats to the ship's offensive power [Ref. 1: p. E8]. Realistic training is currently in the form of self study, periodic shipboard lectures, seminars, drills, or attending one of the Navy's Fire Fighting Schools. Unfortunately, due to a busy ship's schedule and the high rate of shipboard personnel assignment rotations, these forms of training are sometimes unavailable or result in the pass or fail evaluation of the overall damage control supervisor's actions with little time for analyzing the individual team members' actions. A fire team is composed of personnel who are each trained for a specific fire fighting task which requires specific equipment. When a fire team member fails in the execution of his job or the operation of his equipment, the effectiveness of the damage control fire team in controlling a fire casualty is greatly diminished.

To ensure a shipboard damage control fire team can be relied upon, cost and time effective quality training opportunities must be available to ensure individual fire team members gain sufficient expertise at their respective jobs. Private tutoring is generally found to be the most effective form of instruction [Ref. 2: p.1]. Students working with private human tutors learn material up to four times as quickly as those in the typical classroom situation [Ref. 2: p. 1]. They attain a better grasp of the material

than a comparable group of students spending the same amount of time in the classroom [Ref. 2: p. 1]. The US Navy doesn't have the large number of human tutors required to support this fire team member tutoring concept on all of its ships.

This study examines an alternative for the human tutor, namely a computer-based Artificially Intelligent Expert Tutoring System. Programmed with information from US Navy damage control references, it can simulate shipboard fire casualty scenarios. This simulation is in the form of user displayed reports from the fire scene, including fire-related status, action-completed reports, action-incompleted reports, and personnel or equipment casualty reports. Personnel in this environment can be made aware of the overall effects of their seemingly isolated actions. The Tutoring System can ask questions, require responses to a given situation, and then provide the student with immediate results in the form of positive acknowledgement for correct responses or a well formed explanation of why the student's response was incorrect. A computer based Expert Damage Control Tutoring System, available twenty four hours a day on an inexpensive micro-computer, can provide a cost and time effective quality training environment for US Navy fire team personnel.

Numerous expert tutoring systems, also known as Intelligent Computer Aided Instructional(ICAI) Systems, on a wide range of subjects have been developed and are in operation today. SOPHIE teaches problem-solving skills in the context of a simulated electronics laboratory [Ref. 3: p. 247]. In that system, the problem facing the student is to find the faults in a malfunctioning piece of equipment whose characteristics he obtains by taking measurements. SCHOLAR tutors students about simple facts in South American

geography [Ref. 3: p. 236]. WHY tutors students in the causes of rainfall, a complex geophysical process that is a function of many interrelated factors [Ref. 3: p. 242]. WEST is a computer-based learning environment in which the student is involved in an activity, like playing a computer game, and the program operates by "looking over his shoulder" during the game and occasionally offering criticisms or suggestions for improvement [Ref. 3: p. 254]. WUMPUS is a computer game in which a player must track down and slay the vicious Wumpus while avoiding pitfalls that result in certain fictional death. To be a skilled Wumpus-hunter, one must know about logic, probability, decision theory, and geometry [Ref. 3: p. 261]. GUIDON, teaches students diagnostic problem-solving such as medical diagnosis. It contains an interactive dialogue for assisting a user in diagnosing a patient suspected of having an infectious disease [Ref. 4: p. 1]. BUGGY can determine accurately a student's misconceptions(bugs) about basic arithmetic skills. It provides a mechanism for explaining why a student is making an arithmetic mistake, as opposed to simply identifying the mistake [Ref. 3: p. 279]. EXCHECK provides a reactive environment, similar to SOPHIE, to track students progress in formulating arithmetic proofs [Ref. 3: p. 283]. SPIRIT, tutors probability theory, and is a system designed to evolve over time as the theory of student learning evolves [Ref. 5: p. 1]. The GEOMETRY TUTOR teaches high school geometry proofs [Ref. 6: p. 1]. A LISP PROGRAMMING tutor has been developed to help students learn to program in LISP [Ref. 2: p. 1]. All of these ICAI Systems carry on a dialogue with the student and use the student's mistakes to diagnose his misunderstandings.

This study discusses the design issues and the implementation of an ICAI system for tutoring fire team leaders in refining their knowledge and in performing their

duties during simulated casualties. This ICAI system, the Fire Team Leader Learning Center, will henceforth be called "Fire". The research questions which this study addresses are the following:

1. Is it feasible to develop an effective, challenging, expert, fire fighting system program, for the purpose of training fire team leaders in combatting fires on US Naval Ships?
2. Can a rule based expert system, which will correctly analyze symptoms and make proper decisions to extinguish fires, be derived from US Navy Damage Control Training references?

The answers to these questions are pursued in Chapter III during the design and implementation of this system, and Chapter IV's feasibility study. These research questions are answered directly in Chapter V during the final summary.

Nearly everyone who has written on the subject of computer-based education agrees that the potential is enormous. The question now is not <u>whether</u> computers will find a place in education but <u>how</u> [Ref. 7:p. 31]. ICAI tutoring systems, such as Fire, is one feasible method in the US Navy. By providing a much needed one-on-one training environment for the fire teams' personnel, readily available on small micro-computers, the effectiveness and reliability of our Navy's Damage Control Organizations can be assured.

The availability of a computer which can execute the PROLOG computer language is a prerequisite to implementing Fire. Chapter II provides the background on typical damage control training aboard US Navy ships. This background discussion includes present damage control organization training requirements, present methods of performing these requirements, and potential problems with the current training programs. Chapter III provides a complete discussion of the design and implementation of Fire. Chapter IV provides a discussion on the issues of using, refining, and expanding Fire. Chapter V is the Summary and provides a final discussion of the research questions. This

section also discusses some weaknesses with Fire and points out the benefits which the Navy can reap as a result of using ICAI damage control tutoring systems like Fire on its ships. Appendix A contains an exhaustive user session listing demonstrating all of Fire's operations. Appendix B contains a listing of Fire's major supporting files.

II. BACKGROUND

A. SHIPBOARD DAMAGE CONTROL ORGANIZATION FOR FIRE FIGHTING

For the orderly, efficient, and most expeditious employment of manpower and materials to fight a fire, ships survey their own conditions regarding the availability of men and materials. They then assign specific responsibilities, duties and employment, prepare and publish such information in a comprehensive and intelligible form called the "Fire Bill", and make it available to all personnel involved in such activities. The purpose of the Fire Bill is to establish a fire fighting organization and specify certain responsibilities for its direction to ensure that fires in ships are effectively fought and extinguished. While the ship is underway, the Repair Party personnel, who are members of the primary shipboard fire fighting team at sea, report to their General Quarters stations on Fire Call. While the ship is in port, the ship's Fire Bill may designate the Inport Fire Team as the primary fire fighting team. The Inport Fire Team is composed primarily of personnel in the regular damage control repair parties, resulting in each duty section having an effective fire fighting force. [Ref. 8: p. C11]

B. SHIPBOARD DAMAGE CONTROL TRAINING PROGRAM

The organization of a fire fighting team depends on the number of trained men available. A typical Inport Fire Team aboard an aircraft carrier consists of thirty people. A ship typically has six inport duty sections, each requiring its own fire party. This results in 180 personnel required for the Inport Fire Team Organization. These personnel are provided by every division on the ship. They are required to have met the damage control training requirements,

specified in the General Damage Control Qualification Standard required to be completed by all hands within 6 months of reporting aboard. They also must meet the qualification requirements of their specific job on the fire team. [Ref. 8: p. C11]

The qualification programs are administered under the Navy's Personnel Qualification Standards(PQS) Program. These qualification standards require theoretical, operational, and hands on knowledge demonstrations to be witnessed and verified by qualification petty officers who are subject matter experts. The General Damage Control Qualification Standard alone has over 80 knowledge demonstrations required to be performed. The time to train an individual on both General Damage Control and for a specific job on a fire team can take up to 8 months depending on the individual's motivation and learning ability. In a shipboard environment, where personnel's tours of duty assignments rotate frequently, maintaining a stable 180 manned, trained, and fully qualified Inport Fire Team Organization is a big problem.

The very high standard of Navy fire fighter, which the shipboard damage control training programs must produce, is described in [Ref. 8: p. D2],

> The Navy fire fighter acts with a thorough understanding of the means heat is transmitted. He knows all the possibilities in a set of conditions at a fire and, even though he may be unable to confine a fire within bounds, he should not be taken by surprise. He is always prepared to make any rapid adjustment in his extinguishing methods as the changes in the fire boundaries require.

A Navy fire fighter should also "be able to combat, control, and extinguish practically any fire which could be encountered in a ship [Ref. 8: p. C13]. "

The damage control training program is a bilateral system which includes both on and off the ship training.

The training opportunities available off the ship include a division damage control petty officer course, a refresher and advanced fire fighting course, a pipe patching course, and a repair party leader course. Shipboard continuing training programs include periodic "All Hands" basic damage control training, lectures and seminars addressing damage-control-related topics, and realistic damage-control drills monitored by the ship's drill team.

The method and effectiveness of the Inport Fire Team training programs have been a matter of concern for many years. In 1976, Bissel [Ref. 9: pp. 125-126] stated:

> The Inport Fire Party training is as important to the safety of the ship as the training of the underway battle-station repair parties. However, the training of the inport fire party is often perfunctory, consisting of a "muster with equipment" or a "walk through" drill only. Those making assignments to the duty-damage-control parties often consider not the abilities necessary for the tasks assigned, only the requirement to provide a body! The primary training evolution should be an actual drill, emphasizing realistic symptoms, or as a minimum, a formal instruction period on some phase of damage control.

US Naval Ships today typically conduct an actual daily drill or formal instruction for inport fire teams. The difficulties which inherently affect the success of this training is the absence of an effective drill team present after normal working hours to conduct realistic drills, or 100% attendance of team members for lectures when the subject matter doesn't relate to most of their fire team jobs. Again it is apparent that maintaining a stable 180 manned, trained, and fully qualified inport fire team organization is a big problem.

Effective fire teams gain experience and expertise primarily through numerous hours of realistic casualty drill training. This type of training is normally initiated by a knowledgeable drill team. They can supply realistic casualty symptoms(e.g., smoke, darkness, flames, personnel

injuries), challenge fire team members by testing their abilities to carry out their jobs and operate equipment, provide a realistic shipwide casualty environment(e.g., alarms, shipwide announcements), and conduct an effective debriefing to critique each person's actions and evaluate the overall performance of the fire team.

C. ALTERNATIVE TRAINING METHOD FOR INPORT FIRE TEAMS

1. ICAI Inport Fire Team Tutoring Systems

Intelligent Computer Aided Instructional computer programs are a cost-effective substitute for an absent knowledgeable drill team for training inport fire teams. They can provide realistic casualty symptoms, they can challenge each fire team member on an expert level, in both job responsibilities and equipment operations, they can provide a realistic shipboard environment, keep an accurate record of performance scores on all aspects of a casualty, and finally they can effectively tutor a team member when an incorrect action is taken. This study, discusses the design and implementation of an ICAI system for fire team member training. Using ICAI systems for inport fire team training provides, in a one-on-one tutoring environment, both effective casualty strategy training, and specific role domain and equipment knowledge training. Using ICAI systems in this manner, can significantly contribute to solving the problem of maintaining stable 180 manned, trained, and fully qualified inport fire team organizations.

2. Analysis of the Role of a Fire Team Leader

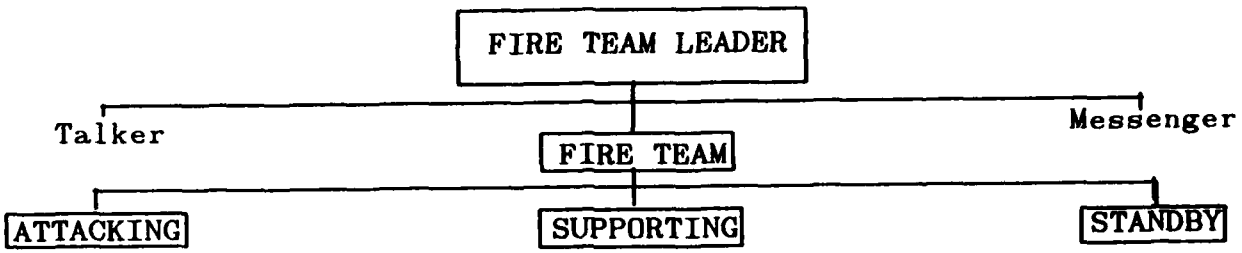
Every inport fire team must have a Fire Team Leader who can effectively coordinate the efforts of his fire team personnel, and who is knowledgeable in all functions and equipment which his team provides. A typical Inport Fire Team Organization is shown in Figure 2.1. Since the fire team leader is the most difficult member to replace, and since all fire team members should be in training to become

a fire team leader, this study focuses on the developments of an ICAI tutoring system for training a fire team leader.

In order to develop an ICAI tutoring system for a specific domain of interest, in this case a fire team leader, all aspects and characteristics of the domain must be well understood. A damage control fire team leader must be capable of making correct decisions during a changing casualty, based on his assessment of the current state of the casualty. Located a short distance from the actual space containing a fire, his assessment must be derived from the reports he receives from the scene.

During actual shipboard casualties, scene reports can often change or contradict previous reports due to unexpected events. These unexpected events include human errors, personnel casualties, fires reflashing after previously being extinguished, and equipment malfunctions. He should take action to recover from personnel injuries, or malfunctioning equipment, before taking new actions to combat the fire. A fire team leader must continually be ranking his concerns as a result of the reports he receives.

Unsatisfactory or incomplete reports from the scene must always be investigated by the fire team leader. If an unsatisfactory report of completion of an order is received, he should order that action again to ensure its satisfactory completion prior to continuing to the next major action. In general, there is a preferred order to complete major actions while combatting a fire. If one action is not completed properly prior to going to the next action, a bad consequence of prematurely continuing may occur. For example, if only 90% of a fire is extinguished and the order to desmoke or remove the smoke from the space is ordered, the high rate of air flow from the desmoking fans will cause the fire to increase in intensity and refill the space. It is imperative that a fire team leader understand the general



ATTACKING	SUPPORTING		STANDBY
Nozzleman	Electrician	Dewatering Team	Nozzleman
Hosemen	Hospital Corpman	Desmoking Team	Hosemen
Investigators	Foam Equipment Operators	Oxygen Tester	Investigators
Plugmen	Hosemen	Explosive Gas Tester	Plugmen
Access Men	Oxyacetylene Cutting Outfit	Firefighting Suit Men	Closure Detail
CO2 Men			CO2 Men

Figure 2.1 Organization of a Large Fire Fighting Team[Ref. 8: p. C13].

order of actions or strategy of combatting a fire and how to effectively concentrate his resources to carry out these actions.

In addition to strategic knowledge, the fire team leader must be an expert in the factual knowledge of his job. The factual knowledge of a fire team leader is understanding every member's job and equipment operations which fall under his control. This is necessary to differentiate between satisfactory and unsatisfactory reports. For example, he must understand the Oxygen Tester's job to recognize a report of 16% oxygen is unsatisfactory for human breathing, whereas 21% is satisfactory. The factual knowledge of a fire team leader covers a wide range of functions and equipments.

The success of a fire team lies heavily on the overall strategic and factual knowledge of the fire team leader. New fire team leaders are often weak in their knowledge of casualty control. This weakness, or inexperience, could result in the failure of the fire team to control a fire. This lack of control may lead to a loss of life or equipment. The shipboard damage control training program, with the aid of ICAI tutoring systems, can assist in ensuring that fire team members and leaders have mastered their strategic and factual knowledge required for their specific jobs.

III. FIRE: A FIRE TEAM LEADER LEARNING CENTER

A. FIRE SYSTEM CONFIGURATION AND DESCRIPTION

Fire is an ICAI tutoring coaching system in which the user is a fire team leader and must direct a shipboard fire team through the actions of extinguishing a fire. The term "coach", describes a computer-based learning environment, in which the student is involved in an activity, such as playing a computer game, and the instructional program operates by "looking over his shoulder" during the game and occasionally offers criticisms or suggestions for improvement without destroying the student's fun at the game [Ref. 3: p. 254]. To be a skilled fire team leader, one must know how to prioritize reports from the scene, be knowledgeable on all operations of fire team equipment and functions which the fire team provides, and be able to effectively respond to unexpected events, such as personnel casualties, broken equipment, or reflashing fires. The user is challenged in these areas in a computer game environment, where his experience level directly affects the fire scenario difficulty. In keeping with the philosophy of computer coaching, students become highly motivated to learn the fundamental fire team leader skills.

The design of the Fire system involves the interactions of the specialist programs shown in Figure 3.1. The system has four special modes of operation. Three of these modes provide individual simulated fire team actions or equipment operations' training, which allows the user to concentrate on his weak knowledge areas. These three modes are the Individual Equipment Operations and Basic Damage Control Actions, the Individual Complex Damage Control Actions, and the Previous Fire Casualty Operations. The fourth mode of operation is the Complex Fire Casualty.

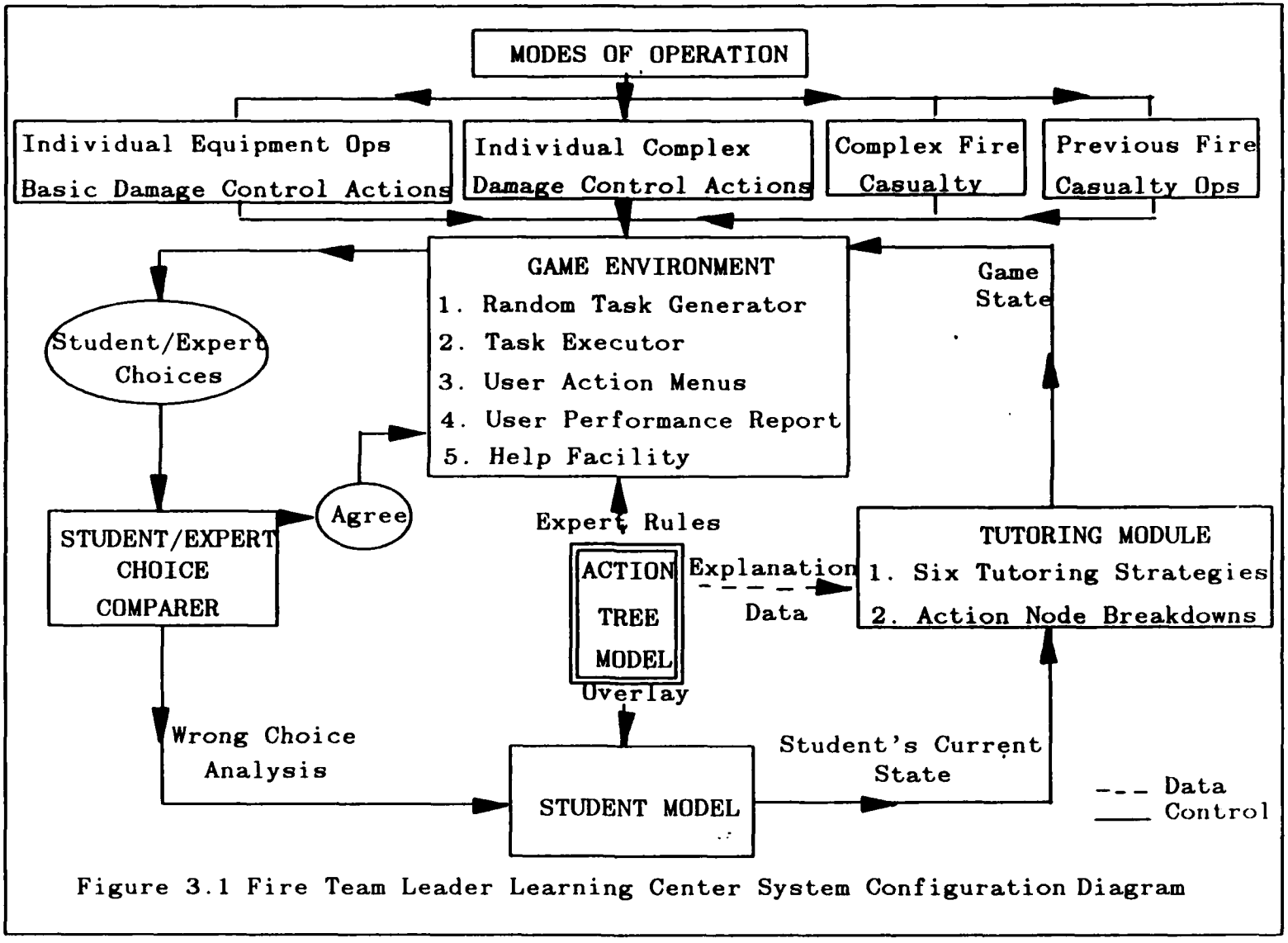


Figure 3.1 Fire Team Leader Learning Center System Configuration Diagram

This mode provides a simulated complex shipboard fire casualty scenario where all damage-control-related actions must be executed correctly to remove the fire from the ship.

The central box of Figure 3.1 contains a tree model representation for an expert fire team leader's actions which are required to correctly execute any fire-related action or equipment operation. It is, in essence, a formal representation of an expert fire team leader's knowledge domain. The tree model utilizes a top-down design, using stepwise refinements of actions with the root as the most general action, to the leaves eight levels below, which are specific action or equipment operations. The Game Environment uses this Action Tree Model to determine the scenario of the simulated fire casualties. The Student Model uses this Action Tree Model to determine the difference between the student's action tree and the expert's. Using this difference, the Student Model focuses on the most important action which the student forgot and communicates this information to the Tutoring Module. The Tutor derives relationships between actions from the Action Tree Model, which can be employed to improve its tutoring explanations, and allows the user opportunities to learn additional details of actions which were forgotten or incorrectly executed.

All of the Fire system's modes of operations use four special modules and the Fire's Action Tree Model to carry out their operations. The four modules are the Game Environment, the Student/Expert Choice-Comparer, the Student Model, and the Tutoring Module. The Game Environment presents the scenario of the fire using random casualties and scene reports, provides user action menus for choosing the next preferred action, maintains the user's performance score, and provides a HELP facility. The Game Environment sends the user's next action choice and the Action Tree

Model expert's next action choice, to the Student/Expert Choice-Comparer to compare the two choices, and return control to the Game Environment if the choices agree. Upon disagreement, the Comparer module sends the user's choice to the Student Model. Using the Action Tree Model and the overlay model method [Ref. 3: p. 231], The Student Model determines how the student's action tree model differs from the expert's Action Tree Model. The Student Model then sends the most important action difference information to the Tutor. The Tutor, basically chooses a tutoring strategy based on the kind of action error, as determined by the Student Model, which the user has made. The Tutor presents an explanation to the user to correct his action error, and also provides the user an opportunity to learn more detailed information concerning his forgotten action or incorrectly executed action.

The Fire system specialist programs, modules, and supporting features shown in Figure 3.1 will be described in complete detail in the rest of Chapter III. The details will provide, where appropriate, a description of the design, the design issues which were addressed, and implementation methods used. The novelty of this Fire system is that in a single system, there is significant fire team leader domain expertise, a broad range of possible interaction strategies available to the tutor, and a modeling capability for the student's current knowledge state. A listing of a user session demonstrating Fire's operation is contained in Appendix A. The Fire system's main program and major supporting procedure files are contained in Appendix B.

B. FIRE SYSTEM DETAILED DESIGN AND DESCRIPTION

1. Action Tree Model

The Fire system's fire fighting related knowledge is stored in a tree data structure. The tree consists of 100

nodes, each representing an action which a fire team leader must used to recover from a shipboard fire. The node structure results from using a top-down stepwise refinement design methodology. The three types of actions nodes which make up the Action Tree Model are Expandable, Non-Expandable, and Test Procedure. The type of node is directly related to the complexity and makeup of the action which the node represents. The nodes each have a unique number which represents their unique location in the Action Tree Model. The structure of the tree and the location of each action's node, imposes a preferred order for action execution which an expert fire team leader would follow during a fire casualty. As a result of each action node's type, location, and relationship to neighboring action nodes, the Action Tree Model is the driving force that effects all aspects of the operation of the Fire system.

a. Action Node Descriptions

(1) Expandable(ex) Action Nodes. An expandable action node in the Action Tree Model corresponds to an action which through stepwise refinements can be broken down into specific subactions which have a preferred order of execution. This results in each generalized high-level action being subdivided into more specific and detailed subactions that, when properly executed, will accomplish the high-level action. If the subactions are also of the expandable type, then their subactions describe in greater detail the lower-level steps needed to accomplish a high-level action. This development in a "treelike" fashion, as shown in Figure 3.2, has a preferred left to right order of actions on all levels. [Ref. 10: p. 165]

In general, there is a preferred order of performing major fire fighting actions. This is the reason the expandable node "treelike" data structure was chosen as the underlying structure for the Action Tree Model. Some

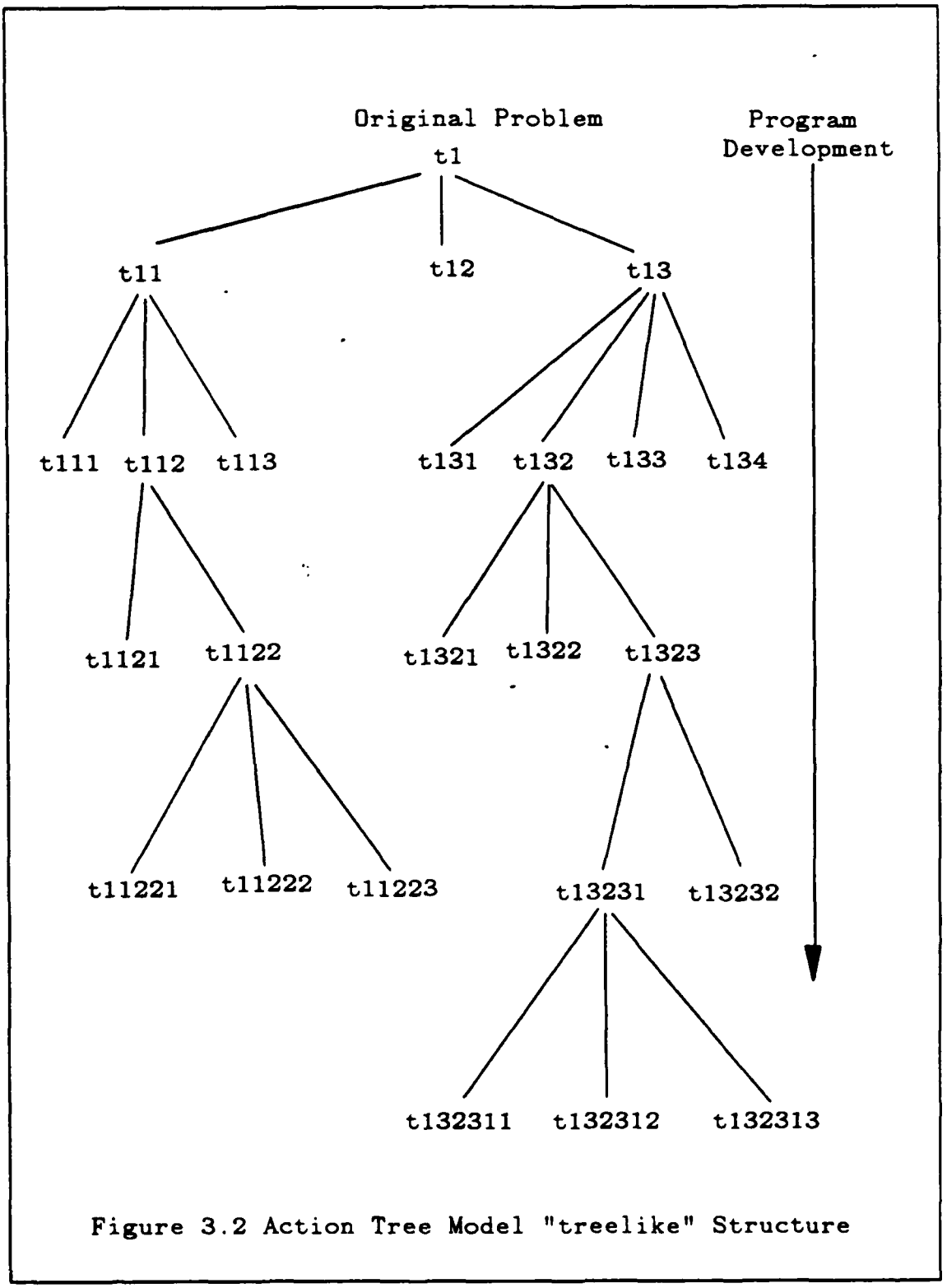


Figure 3.2 Action Tree Model "treelike" Structure

nodes, typically the leaves of the tree data structure, are not expandable. These not expandable node types, will be discussed in section III.B.1.a.(2) and (3). To demonstrate the development of Fire's Action Tree Model, Figure 3.3 shows the root of the tree as the action "Return the ship to normal operation." This action is broken down into subactions which are further broken down. These subactions have a preferred order of execution as reflected in their numbering scheme(e.g., action 1.1.1 is performed prior to action 1.1.2, action 1.1 or its subactions must be completed prior to starting action 1.2 or its subactions). Characteristic of subactions of an expandable node action are the existence of bad consequences which can result when two consecutive subactions are performed out of order.

1. Return the ship to normal operation.
   - 1.1 Put out the fire.
     - 1.1.1 Isolate the fire.
       (Further breakdown of action 1.1.1)
     - 1.1.2 Remove all fires.
       (Further breakdown of action 1.1.2)
   - 1.2 Cleanup after the fire.
     - 1.2.1 Remove all smoke from the space.
       (Further breakdown of action 1.2.1)
     - 1.2.2 Test for sufficient oxygen.
       (Further breakdown of action 1.2.2)
     - 1.2.3 Dewater the space.
       (Further breakdown of action 1.2.3)
     - 1.2.4 Put away the fire fighting Equipment.
       (Further breakdown of action 1.2.4)
     - 1.2.5 Secure the fire team.
       (Further breakdown of action 1.2.5)

Figure 3.3 Breakdown of an Expandable Action Node

The expandable node "treelike" structure also represents the way a human expert fire team leader would approach a fire casualty. Upon hearing the announcement of a fire casualty, a fire team leader would

first think "What is my current most important concern?", it is to "Return the ship to normal operation." But how do I "Return the ship to normal operation," oh by first "Putting out the fire," and then "Cleaning up after the fire." But how do I "Put out the fire?," oh by first "Isolating the fire," and then "Removing all fires" and so on. Having a tutoring system which accurately reflects the way the student must think is critical to the success of an ICAI system. In [Ref. 3: p. 228], this philosophy is supported; "A good teacher must understand what the student is doing, not just what he is supposed to do."

Each expandable node represents a procedural expert in the corresponding subactions that a user must learn in order to acquire the skill in completing the high-level expandable node action. This is accomplished using a separate file of production rules for each expandable node action which acts as a small "expert system" for that node action. The node expert systems will, upon successful completion of a subaction based on an "OK" report from the scene, proceed to its successor subaction as defined in the file's successor rules information table. If a subaction is not completed, a "not OK" report is received from the scene, and that subaction must be performed again. This closely parallels how a fire team leader will act as a result of reports from the fire scene. The node expert system file framework is shown in Figure 3.4.

For each subaction, a source file table contains a number description of the node, its source file name, and its node type. Expandable nodes and test procedure node type nodes actually have a corresponding source file. Non-Expandable node type nodes do not have corresponding source files. A role predicate exists which contains the title of the fire team member which this node action is directed towards. All of Fire's expandable node

```
File Name: action

successor(action, State, Successor):-
     successor_rule_info(action, State, Successor),
     not(result(action, State, nok, _).

successor_rule_info(action,{1,...,0},{1,...,1}).
successor_rule_info(action,{1,...,1},{1,...,2}).
                        .
                        .
                        .
successor_rule_info(action,{1,...,X},complete).

source file(action,{1,...,1},node file name,node type).
                        .
                        .
                        .
source file(action,{1,...,X},node file name,node type).

role(action,'FIRE TEAM LEADER').
start(action,{1,...,0}).

/** Main Fire Program contains the following rule.**/
successor(_,State,State).
```

Figure 3.4 Expandable Node Action File Framework

action roles are for the fire team leader, but they could be easily changed to any other member's role which the user must assume(e.g., medical corpman for the personnel casualty actions). Finally, a start predicate is required, whose dummy node number's successor is the firest subaction which will be performed. This initiates the correct execution of the remaining subactions in an order prescribed by the random report results received from the scene. With these numerous fire team member action procedural experts, the overall Action Tree Model assumes the expertise required of an expert fire team leader.

(2) Non-Expandable(nex) Action Nodes. A non-expandable action node is the simplest of the node types. It corresponds to an action which does not breakdown into specific subactions or requires any supporting information. The non-expandable action node is always a

leaf on the Action Tree Model. It doesn't have a corresponding source file. It is always a subaction of an expandable node action as shown in Figure 3.5.

1.1.1.1.3.3. Relieve the senior person in charge at the scene.(ex)

1.1.1.1.3.3.1. When ready to receive briefing from the man in charge state to him, "I am ready to relieve you."(nex)

1.1.1.1.3.3.2. Receive briefing from man in charge. (Obtain who, what, when, where, and how, information about the casualty).(nex)

1.1.1.1.3.3.3. When ready to take charge of the scene state, "I relieve you."(nex)

1.1.1.1.3.3.4. Announce to all personnel at the scene that you are the man in charge.(nex)

Figure 3.5 Breakdown of an Expandable Action Node to Non-Expandable Action Nodes

(3) Test Procedure(tp) Action Nodes. A test procedure node in the Action Tree Model corresponds to a basic action which doesn't breakdown like an expandable action node, but can be further described by supporting information. This supporting information can be in the form of a basic, easy to remember, step by step procedure, such as taking an oxygen test with a gas tester. This type of test procedure node will be said to contain an "order-dependent" description. The other form of supporting information is amplifying information such as guidelines, advice, or safety precautions which should be followed during the execution of the test procedure node action. This type of test procedure node doesn't contain order-dependent information and will be said to contain an "informational" description. Test procedure nodes will always be leaf nodes on the Action Tree Model. They will

also always be subactions of expandable node actions as shown in Figure 3.6.

```
1.2.2. Test for sufficient oxygen.(ex)
  1.2.2.1. Get the oxygen test equipment.(nex)
  1.2.2.2. Test the oxygen test equipment.(tp)
  1.2.2.3. Obtain a satisfactory first oxygen test.(tp)
  1.2.2.4. Obtain a satisfactory second oxygen test.(tp)
```

Figure 3.6 Breakdown of an Expandable Action Node to Test Procedure Action Nodes

As previously discussed, order-dependent description test procedure action nodes, usually represent basic equipment operational procedures. Characteristic of these procedures are an order-dependent step-by-step procedure. A fire team leader must know these procedures, generally through memorization, so that the equipment will operate properly. These node's subactions differ from expandable node's subactions, in that, if they are performed out of order, the equipment being operated will not work properly, whereas expandable node subactions performed out of order may cause a potentially catastrophic bad consequence. These node action file's framework, shown in Figure 3.7, involve a series of operation steps listed in correct operational order, the action or task name which is also the file's name, and finally the reference which this action procedure was derived. In Figure 3.6, action "1.2.2.2. Test the oxygen test equipment." is an order-dependent description test procedure node.

Also as previously discussed, the informational description test procedure action nodes, usually represent non-order-dependent amplifying

```
File Name: action_name
operation_step({'first action procedure step'}).
                :
                :
operation_step({'last action procedure step'}).
task_name(action_name).
reference(reference_name).
```

Figure 3.7 Order Dependent Description Test Procedure Action File Framework

information. This information is generally in the form of guidelines, advice, or safety precautions which should be followed during the execution of these test procedure node's action. These node action file's framework, shown in Figure 3.8, is similar to Figure 3.7 but contains only one operation step predicate containing the entire text of amplifying information.

```
File Name: action_name
operation_step({'  Amplifying Information  ',
                '  Body of Text            ',
                '  End of Text             '}).
task_name(action_name).
reference(reference_name).
```

Figure 3.8 Informational Description Test Procedure Node Action File Framework

In Figure 3.6, action 1.2.2.3. and 1.2.2.4. are both informational description test procedure node actions.

b. Top-Down Design

The top-down design methodology used to design the Action Tree Model begins with the overall goal of a fire team leader, that is, to restore the ship to normal operation. Then a series of stepwise refinements were applied to develop a large tree model with a maximum depth of eight levels in some parts of the tree structure. The third level actions, being the more general descriptions of actions, represents a basic beginner's level of knowledge of a fire casualty:

1. Isolate the fire.
2. Remove all fires.
3. Remove all smoke from the space.
4. Test for sufficient oxygen.
5. Dewater the space.
6. Put away equipment.
7. Secure the fire team

As the action's are broken down into further subactions, each with important order dependence, the levels assume the "Advanced" and "Expert" knowledge levels of detailed fire fighting knowledge. As stated in [Ref. 11: p. 106], "the interesting feature of top-down refinement is the flexibility of the abstractions. Abstraction states are individually constructed to fit each problem in the domain." This is evident from Figure 3.3 where the abstract action 1.1 is broken down into two subactions and action 1.2 is broken down into five subactions.

The top-down design approach was instrumental in the transition from design to program implementation. As in structured programming, stubs [Ref. 10: p.167], were used for subactions which did not yet exist during the early stages of program implementation. This allowed the upper levels of the Action Tree Model to be tested, prior to the completion of the lower levels.

c. Action Node-Numbering System Description And Benefits

The Action Tree Model consists of a hundred action nodes. Each action node has a unique number based on its location in the tree. The root of the tree is node number one and has the Prolog list representation of "{1}". The subactions of the root have numbers 1.1 and 1.2 and are represented in Prolog list format as "{1,1}" and "{1,2}" respectively. Assigning node numbers to actions results in the following benefits:

1. Less programming errors due to easy-to-develop short numbers compared to numerous copies of action description character strings which are proned to misspelled words, extra spaces, or incorrect use of apostrophe's in a Prolog environment.
2. Using list processing rules, the node position location in the Action Tree Model can easily be determined. This information is critical in comparing a user's answer to the expert's answer and formulating the correct student knowledge model tutoring strategy.
3. The node action's location is also useful in displaying a specific action node's subactions inorder to teach a user a specific action.
4. The numerical value of the node number also dictates successor action relationships used in the overall action order control of the system game.

The list format of the node number allows outstanding flexibility for the Action Tree Model for future changes or expansions. The current version of the Tree Model has a root number of "{1}" and contains an action node number "{1,1,1,1,1,3,1,2}" on the lowest level representing the action "Have personnel turn their helmet lights on." This particular action node is a non-expandable type node. But if next year, a complex two step procedure is developed for turning the helmet light on, then the easy modification to incorporate this change is to classify "{1,1,1,1,1,3,1,2}" as an expandable node type, designate two new action nodes "{1,1,1,1,1,3,1,2,1}" and "{1,1,1,1,1,3,1,2,2}" and develop them based on their action node types. There are no other program coding changes to the Fire system for this action node expansion. If a new

type of explosive gas tester is developed and required to be used, then just change ONLY the test procedure files for nodes "{1,2,1,1,2}"(Have the explosive gas tester tested.) and node "{1,2,1,1,3}"(Sample the space which had the fire for explosive gases.). There are no other program coding changes to the Fire system for this explosive gas tester change. The flexible node numbering system in addition to the fixed format of each node type source file allows easy, fast turn around maintenance periods in keeping the Action Tree Model's fire fighting knowledge current. Figure 3.9 is a partial display of the Action Model Tree with all of its supporting features. A Casualty Tree Model, shown in Figure 3.10, identical in structure to the Action Tree Model, implements the fire reflash, personnel casualty, and equipment failure casualties. The casualty scenario traverses both trees in a sequence directed by the Random Task Generator described in section III.B.2.a.

2. Game Environment

The Game Environment's purpose is to maintain the user's interest and motivate the young(generally 18-24 year old) fire team leaders, to use the Fire system. As in West and Wumpus, it also coordinates the user's presentation of a game scenario. In Fire, this is a realistic shipboard fire casualty scenario which is commensurate with the user's experience level. It randomly chooses scene reports based on the user's past performance and on his experience level to challenge potential weak knowledge areas. It provides challenging action menu's of variable lengths based on the user's experience level. A final performance report on all major fire fighting tasks which the user performed is displayed upon completion of the fire casualty scenario or upon exiting of the Fire system.

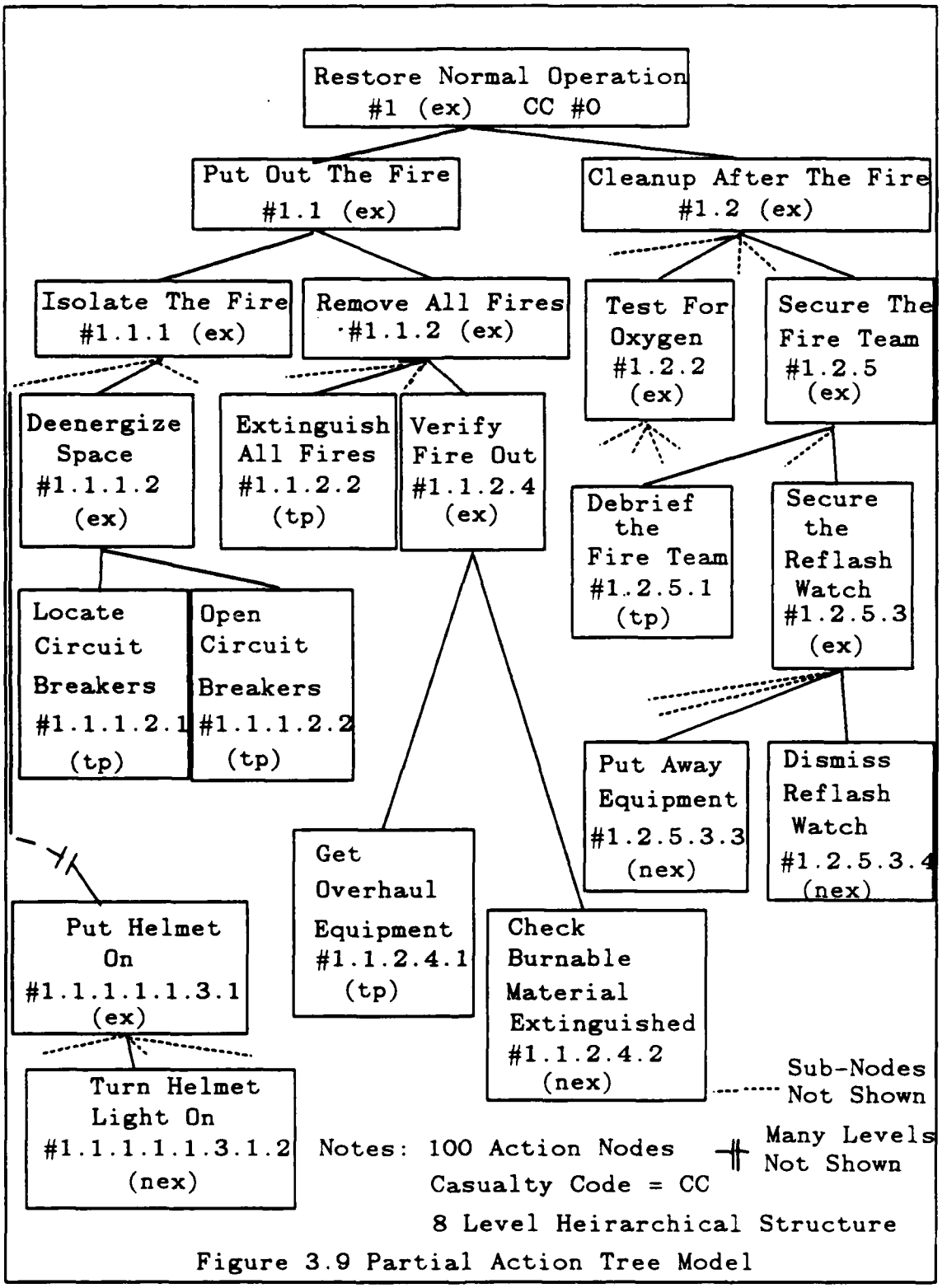


Figure 3.9 Partial Action Tree Model

a. Random Task Generator(RTG)

The purpose of the Random Task Generator is to generate, in a pseudo random fashion, a control mechanism for initiating unexpected events, such as personnel casualties, broken equipment, and fire reflashes. It also controls when a report from the scene reflects when a previous action wasn't completely accomplished. Its final purpose is to determine, based on a user's past performance on a task and his experience level, whether the user will be tested on the subactions of a higher-level task. This unanticipated directing of the specific subactions of a specific task, falls in the category of an unexpected event for a fire team leader. This Random Task Generator adds powerful realism to the shipboard fire casualty scenario. The user, just as a real fire team leader, can not anticipate every future action. He must continually analyze the reports he receives from the scene, rank his concerns, and when directed, carry out the detailed subactions of some tasks as would be required if a fire team member failed in his role responsibilities.

The initiation of unexpected events is a function of the user's experience level, his past performance on the specific unexpected event, and a random number in the range of one to a hundred. Each experience level is assigned a number which defines the number range or "unexpected event window", between it and a hundred where a random number must fall to cause the unexpected event to be executed. The experience levels, number assigned, and corresponding percent of the time unexpected events occur are:

1.	Beginner	90	10%
2.	Advanced	75	25%
3.	Expert	60	40%

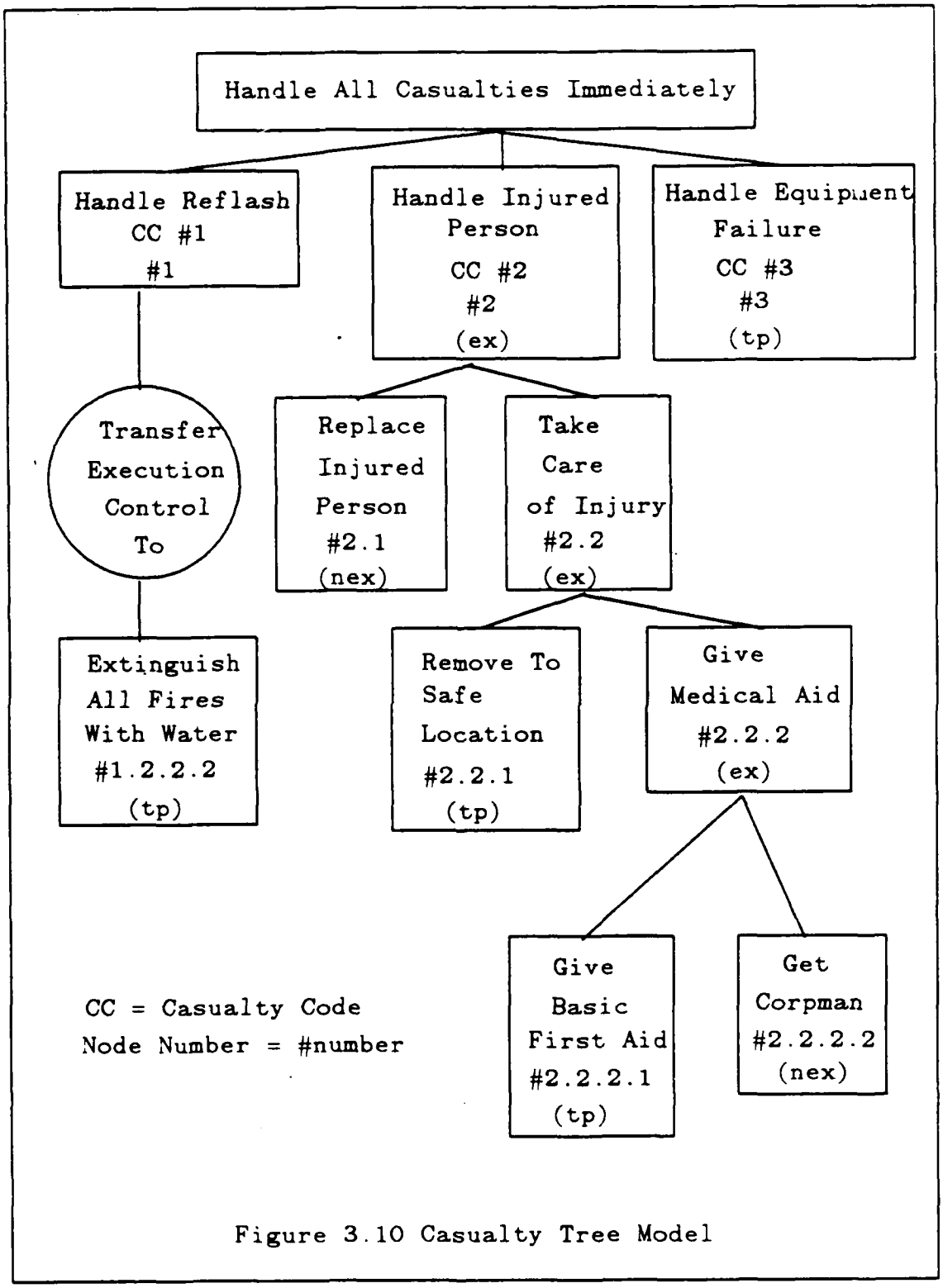


Figure 3.10 Casualty Tree Model

This approach causes the scenario difficulty factor to increase as the user's experience level increases. An expert will receive 30% more casualties and unexpected events than a beginner.

The unexpected event window can be enlarged or narrowed based on the user's past performance on a given task. If the user has a current performance score of 90% or better, for a specific unexpected event, then a -15 points is subtracted from the basic experience level number, which will narrow the window. In this case, the experience levels, numbers assigned, and corresponding percent of the time a particular unexpected event, which the user has performed well on, are:

1.	Beginner	105	0%
2.	Advanced	90	10%
3.	Expert	75	25%

This approach results in the Fire system spending less time on the unexpected events that the user has already performed well on.

The unexpected event window is enlarged for tasks which have a previous performance of less than 90%. For these tasks, the number of points which were previously missed performing this task, are subtracted from the basic experience number up to a maximum of 25. In this case, the experience levels, maximum numbers assigned, and corresponding percent of the time a particular unexpected event, which the user has done poorly on, are:

1.	Beginner	65	35%
2.	Advanced	50	50%
3.	Expert	35	65%

This approach results in the Fire system spending a greater percent of its time on unexpected events that the user has performed poorly on. This repetition of poorly performed tasks allows the user to "learn by doing" the task over and

over until his performance improves and consequently the unexpected event window will narrow and the event will then occur less frequently.

The control of personnel, broken equipment, and fire reflash casualties is described in section III.B.2.b. In controlling the reports from the scene, in order to provide some not fully completed action reports, a biased random number generator is used. For every node action in the Action Tree Model, an action result predicate exists which contains "OK", or action completed reports, and "NOK", or action not fully completed reports. One example of this predicate is shown in Figure 3.11.

```
Action Node: node_action([1,2,2,3],
  ['Obtain a satisfactory first oxygen test.'],tp).

/* Two OK reports exists.  The RTG also determines
   which of the two reports to display when an OK
   report is chosen. */
action_results(mainfire,[1,2,2,3],ok,
   [['The first oxygen test has been completed
      satisfactory.'],['The first oxygen test
      indicates 21% oxygen.']]).

/* Two NOK reports exists.  The first would test
   his specific knowledge of oxygen test results
   the other on his alertness on scene report
   content. */
action_results(mainfire,[1,2,2,3],nok,
   [['The first oxygen test indicated 16% oxygen.'],
    ['The first oxygen test has not yet been
      performed.']]).

/* Node action casualty related information is also
   contained in this predicate. */
action_results(mainfire,[1,2,2,3],[3],
   [['The oxygen tester's meter needle is missing.'],
    ['The oxygen tester's sample hose is damaged.']]).
```

Figure 3.11 Node Action Scene Report Predicate

The biased random generator forces an "OK" report to be displayed to the user two of every three times on the average. This allows just enough "NOK" reports to the scene

leader to test his awareness to scene reports, and his specific fire fighting knowledge for recognizing when an action hasn't been completed. If this random number generator isn't biased, the "NOK" reports "seem" to appear almost every time and the the user gets into a routine of almost immediately reordering every action without really analyzing the reports. This really lengthens the game time and distracts the user from enjoying the game environment.

Since the Random Task Generator directly effects the scenario of the fire casualty, in order to receive a different scenario every time a user plays, true randomness properties must be achieved. These characteristics are approached in two ways. First, as shown in Figure 3.12 the user is required to put in the time of day upon beginning the program. This time of day, a number between 1 and 2400, is used to initialize or "seed" the Random Task Generator. This will result in the beginning of the scenario to be different for different times of the day.

```
/***  Fire System Welcome Message Beginning  ***/
Welcome, I am an EXPERT in FIRE FIGHTING and DAMAGE
CONTROL!  I am ready to challenge your knowledge as
a fire team scene leader or supervisor!
Please enter your NAME:  User Name
User Name, please enter the time of day(0001-2400): 0830
Thank You.
```

Figure 3.12 Initial Seeding of the Random Task Generator Using the Time of Day

Secondly, to prevent the user from receiving the same scenario by entering the exact same time of day every time he uses the program, the Random Task Generator inherits an unpredictable or random characteristic property by changing

the seed every time the Tutoring Module is used. The exact times when a user will make incorrect actions during the scenario is unpredictable. The Tutoring Module is used only when incorrect actions are entered by the user. Therefore, these properties of the Random Task Generator ensures that the user is effectively challenged in an unpredictable environment and developes expertise in the domain of fire team leader fire fighting knowledge.

b. Task Executor

The purpose of the Task Executor is to direct the user interface presentation of the subactions of a designated high-level fire fighting action or task, inorder to thoroughly test the user's knowledge corresponding to this task. The interface presentation includes reports from the scene, action menus of variable length, and requests for user's interactive action inputs. The Task Executor can execute both expandable node actions and test procedure node actions. The user's input, along with the Action Tree Model expert's choice, of the the correct next action, are both sent to the Student/Expert Choice-Comparer module for deliberation.

Expandable node actions are executed by loading in their respective source file containing successor rule information. The decision on the next best action to take by this procedural expert is based on the last "OK" or "NOK" or casualty report received. The user is also tested in this specific situation. The user's choice of action is compared with the Action Tree Model expert's choice, and if correct, the user is given credit and the game continues, otherwise the user takes a "trip" to the Tutoring Module to learn why he was incorrect. This use of "immediate feedback" for incorrect user choices will be described in section III.B.5.

Upon returning from the Tutoring Module, the Task Executor will call upon the Random Task Generator to

decide if a user should be forced to perform the subactions of the particular subaction of the task which is currently being performed. As previously discussed, this depends largely on the user's experience level, and his past performance on this particular subaction. If he is directed to perform the subactions of a particular task subaction, shown in Figure 3.13, a recursive call to the Task Executor will be made passing it the particular subaction which will become the new task to be executed. These recursive calls can continue to all levels of the Action Tree Model.

```
                                        User Name's Score Level
                                              3 of 6      50%
User Name, you are the "FIRE TEAM LEADER".

FIRE TEAM LEADER, you MUST now DIRECT THE ACTIONS
to complete the following TASK: Set fire boundaries.

FIRE TEAM LEADER, indicate your most important,
                  or next action to be completed.

1. Send fire boundary teams to each adjacent
   space with portable fire equipment in order
   to verify the boundaries are containing the fire.
2. Verify all fires are out.
3. Conduct a debriefing of the fire casualty
   with the fire team.
4. Send fire boundary teams to shut all physical
   openings to the space with the fire.
5. Lead the fire team to the Repair Locker.
6. Have Repair Locker locate on ship's compartment
   drawings all adjacent spaces and all physical
   openings to the space including doors, ventilation
   ducts, and drains.

7. HELP
8. QUIT

: 6
```

Figure 3.13 Display of Task Executor Directing User to Perform a Particular Task

Test procedure node actions are executed using a different method as a result of their quite different subaction descriptions, as described in section

III.B.1.a.(3). The order-dependent test procedure node actions, contain easy, step-by-step, fire fighting procedures, which should be simple tasks for a fire team leader. They should be second nature to a fire team leader such as the sequence of steps to start an oxygen breathing apparatus, or how to test an oxygen tester. To test the fire team leader's knowledge, the list of operational steps from the task's source file are randomly mixed up twice, this results in two mixed up lists of steps and the original correct list of operational steps. These three lists are presented to the user one at a time. If he recognizes the correct list, he is awarded a point for the task and the Fire system game continues. Otherwise he is informed that he is incorrect, and the correct order is displayed for him to learn before he continues with the Fire system game. An example of this procedure is shown in Figure 3.14.

The informational test procedure node actions, contain guidelines, advice, or safety precautions which should be followed while performing the node action. Since this information is often "nice to know" or "philosophical", there is no absolute requirement that a fire team leader must follow it. Yet, in general, it will aid him in correctly and safely carrying out a specific node action. Therefore, upon executing these test procedures files, their textual content is displayed for the user to review and learn before he continues with the Fire system game. Their is no test or grade associated with these procedures. An example of this procedure is shown in Figure 3.15.

When a particular task subaction or its subactions have been completed, a node action related casualty may occur. The majority of the Action Tree Model node's actions have a casualty associated with them. The three types of possible casualties, with their corresponding casualty number code, are:

User Name, are the following steps the correct sequence to perform the task:
Test the oxygen test equipment.

1. Pull the knob down and adjust the flame to approximately 3/8 inch in height. Allow the flame to burn about 5 minutes to reach its normal operating temperature.
2. Blow against the flame and gaskets to test the lamp for leakage.
3. Push on the relighter handle(igniter) and slide it up as far as it will go. Turn the relighter handle and ignite the wick.
4. Turn up the wick using the wick adjuster.

Enter a "yes." or "no.": yes.

User Name, you are incorrect, the above sequence is not correct! Below is the correct sequence, STUDY AND LEARN it before you continue.

1. Turn up the wick using the wick adjuster.
2. Push on the relighter handle(igniter) and slide it up as far as it will go. Turn the relighter handle and ignite the wick.
3. Pull the knob down and adjust the flame to approximately 3/8 inch in height. Allow the flame to burn about 5 minutes to reach its normal operating temperature.
4. Blow against the glass and gaskets to test the lamp for leakage. Leakage will cause the flame to flicker.

Figure 3.14 Display of an Order Dependent Test Procedure Action Node

1. Reflash of the Fire.
2. Personnel Injury.
3. Broken or Malfunctioning Equipment

The node's casualty code and associated scene reports are contained in the "action_results" predicate shown in Figure 3.10. The Random Task Generator will determine whether to initiate one of these casualty tasks. This initiation would be done by making a recursive call on the Task Executor with the casualty name as the new task to be executed, otherwise the next task subaction would be chosen based on 'the last scene reports. Example scene reports for the Fire system casualties are shown in Figure 3.15.

* * * KNOWLEDGE REVIEW * * *
User Name, you will be shown a review of important knowledge required in the performance of Task:

Obtain a satisfactory first Oxygen Test.

STUDY AND LEARN THIS INFORMATION and for further guidance refer to reference:
FLAME SAFETY LAMP, NAVEDTRA 465-08-00-82, p. 11.

ANALYZING THE OXYGEN TEST SAMPLE
If the flame grows dim, the oxygen content is lower then normal. Access to the space is permitted only under conditions of extreme need, and then only for limited periods of time. No routine work is to be conducted under this condition. If the fame dies out, there is less than 16 percent oxygen by volume. If the flame flares up brightly, there is a lean concentration of explosive gases or vapors. If the flame goes out with a slight pop, there is an explosive concentration of gases or vapors. If the flame flares up and then goes out, there is a rich concentration of explosive gases or vapors. If no explosive gases are present and there is sufficient oxygen to sustain life, work in the compartment may proceed.

Figure 3.15 Display of an Informational Test Procedure Action Node

FIRE TEAM LEADER, you THEN receive the following report: Desmoking has stopped due to large flames in the space.

or

Large flames have been reported in the vicinity of the Reflash Watch.

or

The overhaul equipment shovel has broken.

or

The explosive gas tester meter has no needle.

or

A man has fallen and is unconscious while trying to close a ventilation duct damper.

or

The electrician was shocked at the electrical panel.

Figure 3.16 Examples of Casualty Scene Reports

c. User Action Menus

As the Fire system Task Executor traverses through the Action Tree Model, which formulates a unique casualty scenario, the user must choose a correct "next action" from a menu of fire fighting actions for every situation. In order to add a greater degree of difficulty as the user experience level increases, the number of menu action items presented are:

1. Beginner 3
2. Advanced 6
3. Expert 12

The menu items are made up of the correct next action, as chosen by the Action Tree Model expert, a nearly equal number of actions which come directly before and directly after the correct action in the Action Tree Model, and when required, "fill in action items" randomly chosen from the fourth level of the Action Tree Model . Listing the menu in this way focuses the attention on all the specific actions which are closely related to the correct next action. The fire team leader will not be able to ignore obviously wrong actions and easily choose the correct next action because there will generally be several listed actions which should be performed during the current time frame. This approach forces the user to rank his present concerns and learn the fine, important, order-related differences between actions. A "HELP FACILITY" and a program "QUIT" feature are also listed at the end of the menu available to the user at all times. An example of the user action menu is shown in Figure 3.13.

d. User Performance Report

The Fire system's Complex Fire Casualty mode presents a realistic presentation of a complete shipboard fire scenario consisting of numerous events or tasks. A

goal of a fire team leader is to successfully recover from a fire casualty in the most effective manner. An ICAI tutoring system, which provides only an overall running score of performance, is a poor evaluator of the potential of a future fire team leader. If user no. 1 performs ten tasks to complete a fire, each with a performance score of 70%, then his overall performance score would be 70%. But if user no. 2 performs the same ten tasks, and his scores are 90% on seven tasks, with the remainder tasks' scores of 25%, 25%, and 20%, then his overall score is also 70%. But clearly, user no. 2 has a much greater potential to be a good fire team leader for the following reasons.

1. User no. 2 has demonstrated that he is capable of achieving outstanding performance grades of 90% on seven of the fire fighting actions. Fire Fighting tasks concepts are usually equal in difficulty in learning. User no. 2, given some more time, has the capability, to master the three other actions and become an effective Fire Team Leader.

2. User no. 1 either doesn't have the intelligence capability to master the ten fire fighting tasks, or as the number of tasks increases, user no. 1 has difficulty in controlling the sequence in which the actions are performed. Both of these characteristics indicate that user no. 1 currently has poor potential to be an effective Fire Team Leader.

A tutoring system needs to know how the student has done with respect to each sub-skill involved in problem solving [Ref. 5: p. 22].

The Fire System provides a Performance Report, shown in Figure 3.17, which gives the overall score and all of the individual fire fighting task scores which were tested. Using this report, the user can identify his specific weak areas and using the alternate training modes of the Fire system, he can train specifically on these tasks. This report is presented upon completion of the complex fire casualty or when a user exits the program using the QUIT action menu option.

```
                    FINAL GRADE OF PERFORMANCE

Congratulations User Name you've survived!

Remember there's always room for improvement in Damage
Control Casualty Training.  Your life, your shipmate's,
and your ship's survival may someday depend on your
KNOWLEDGE as a MAN IN CHARGE  of a casualty.  The
following is a report of your performance:

Casualty    Task          Points Total Percent Grade
Overall Score               36     47      76

Broken Equipment Casualty
Pts = 7  Total = 9  Percent Grade = 78

Extinguish all the fires
Pts = 1  Total = 1  Percent Grade = 100

Have a smoke removal path established.
Pts = 1  Total = 1  Percent Grade = 100

Desmoke the space.
Pts = 3  Total = 3  Percent Grade = 100

Injured Person Casualty.
Pts = 6  Total = 6   Percent Grade = 100

Main Fire
Pts = 18 Total = 27  Percent Grade = 66.6

For further training on the types of damage control
actions or equipment operations you just directed:
Enter a 1. To Quit
       2. Previous Fire Casualty Operations Training
```

Figure 3.17 User Performance Report

e. Help Facility

A fire system Help Facility is always available to the user as an item on the action menu. Its purpose is to provide a brief description of the modes of operation, the standard user menu and performance score, the operation of the Tutoring Module, the effect of the user's experience level on the difficulty of the scenario, and the QUIT option of exiting the system. In keeping with the philosophy of realism in training a user to be a fire team leader controlling an actual casualty, the user must be forced to know exactly what he is doing with little time to ask for

"help." This concept is enforced by removing one point from the user's overall score whenever the Help Facility is used. The introduction to the Help Facility is shown in Figure 3.18.

DAMAGE CONTROL FIRE TEAM LEADER LEARNING CENTER
HELP FACILITY

This "LEARNING CENTER" was developed by LT Steve Weingart USN as a research project in developing an expert system tutorial program associated with the instruction of shipboard fire team leaders. The major goal of this system is to train a fire team leader to direct a fire team's actions in combatting a fire in a computer simulated shipboard casualty environment. The LEARNING CENTER's major training mode of operation is the Complex Fire Casualty Mode, which places the user as the fire team leader in a simulated shipboard fire casualty environment. He/She will have to respond to various reports from the scene which, as in an actual casualty, are sometimes contradictory, inaccurate, or complete, or contain new information on new casualties, such as broken equipment, injured personnel, or reflashes of fires.

Figure 3.18 Help Facility Introduction

3. Student/Expert Choice-Comparer

The purpose of the Student/Expert Choice-Comparer is to perform a simple comparison of the chosen action node numbers. If the node numbers agree, then the user is awarded credit for a correct answer and the Fire system control returns to the Game Environment to continue the casualty scenario. If the node numbers disagree, the user's node action number is sent to the Student Model module for analysis for follow on tutoring.

4. Student Model

A detailed model of a student's knowledge that indicates his or her misconceptions is important for successful tutoring [Ref. 3: p. 279].

The student modeling module represents the student's understanding of the material to be taught. Much recent

> ICAI research has focused on this component. The purpose of modeling the student is to make hypotheses about his misconceptions and suboptimal performance strategies so that the tutoring module can point them out, indicate why they are wrong, and suggest corrections.

[Ref. 3: p. 231] Most current ICAI tutoring systems have a student model module for detecting student misconceptions. The Fire system has a student model module using the form of an "overlay model" [Ref. 3: p. 231].

In an overlay model, the student's understanding is represented completely in terms of the expertise component of the program. This determines where the student's knowledge and the expert's knowledge differs. The analysis of this difference, is the key to determining the best method of tutoring the student back to an expert's level of knowledge. Other current ICAI tutoring systems using the overlay model include GUIDON [Ref. 3: p. 267], and WEST [Ref. 3: p. 254]. The Fire system's expert fire fighting knowledge is the complex structure of node actions and their respective files contained in the Action Tree Model. Conceptually, a non-expert student's knowledge tree, shown in Figure 3.19, looks identical to the Action Tree Model except some action nodes are missing, out of position, or contain incomplete or inaccurate procedural knowledge. By "overlaying" this imperfect student tree model over the Fire system's Action Tree Model, the differences requiring correction by a tutor are evident.

The Fire system, as a result of the flexible Action Tree Model and node numbering system, can easily determine the student's knowledge model using only the previously completed action, the user's wrong choice, and the active casualty code. Since the Fire system uses immediate feedback upon an incorrect answer, it is assumed that the user has performed or been tutored to an expert's level of performance up to the previous correct chosen action. Using

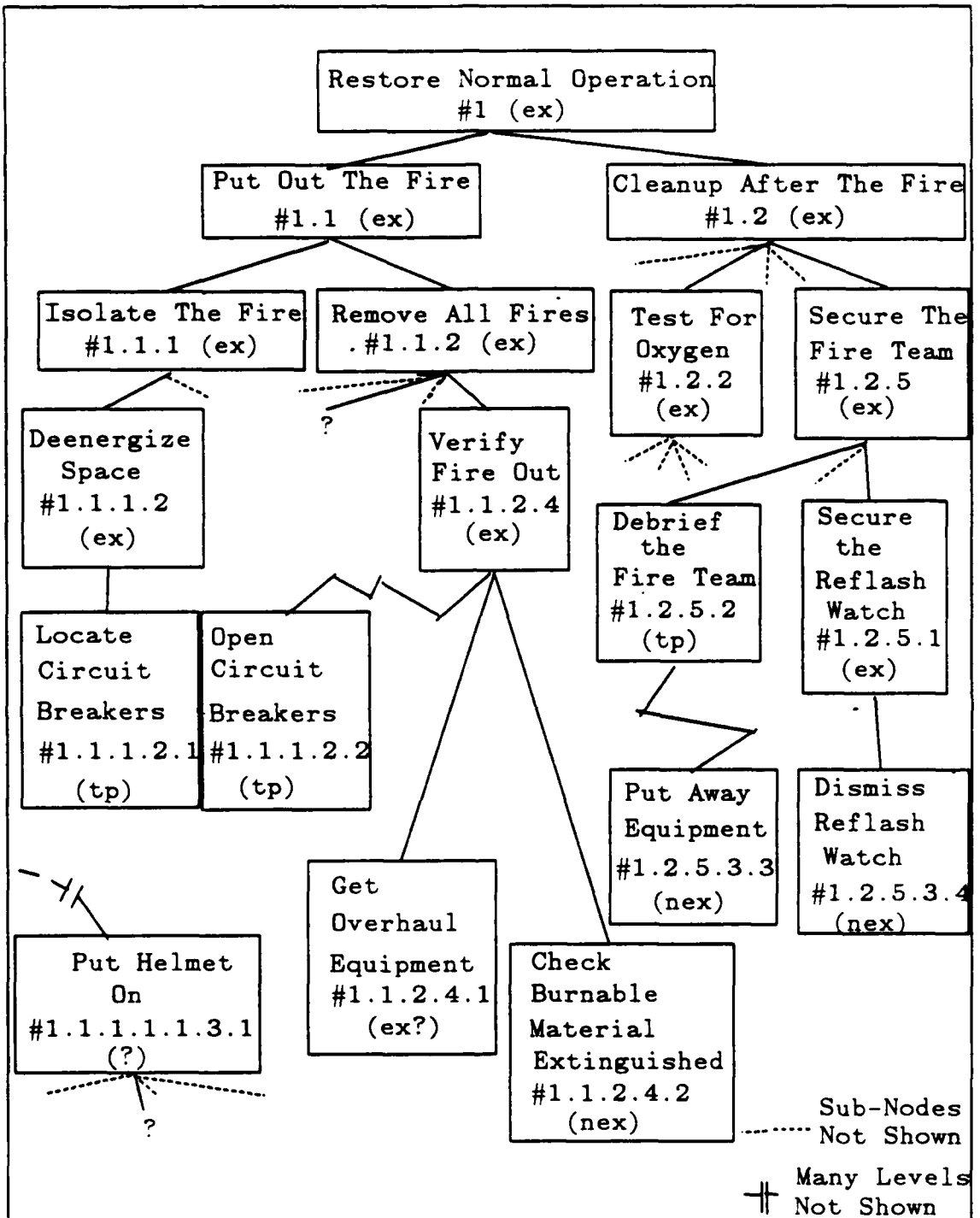


Figure 3.19 Student's Knowledge Tree Model

the relationship between the previous action node and the user's present incorrect action node, and the active casualty code, the following six different student model error cases can be detected.

1. Wrong Task--When an Injured Personnel, Broken Equipment, or Fire Reflash casualty task has been activated, as evident by a nonzero casualty code, the fire casualty scenario traverses the Casualty Tree Model shown in Figure 3.10. A Wrong Task error occurs in this situation, when a user chooses an action on the Action Tree Model shown in Figure 3.9, which relates to the task of "Returning the ship to normal operation", rather then the activated casualty.

2. Weak Casualty Knowledge--When an Injured Personnel or Broken Equipment casualty task is active, as evident by a casualty code of two or three, the fire casualty scenario traverses the casualty code two or three's subtree of the Casualty Tree Model shown in Figure 3.10. A Weak Casualty Knowledge error occurs when a user choses the wrong casualty control recovery action (i.e. chooses action node no. 2.2 before action node no. 2.1).

3. Forgetting Completed Action--The fire casualty scenario traverses the Action Tree Model, when the casualty code is zero. A Forgetting Completed Action error occurs when a user chooses an action, other then the previously completed action, which has already been completed. For example, this error would occur if the user had just completed the action no. 1.2.5.3.3 Put Away Equipment, and then chooses no. 1.1.1.2 Deenergize Space, or action no. 1.1.1.1.1.3.1 Put On Helmet, rather then the correct choice of action no. 1.2.5.3.4 Dismiss Reflash Watch. Using the Action Tree Model, this error is detected when a user chooses any node action to the left of the previously completed action node or to the left of any of its ancestors.

4. Unsure Action Knowledge--In both the Action Tree Model and the Casualty Tree Model shown in Figures 3.9 and 3.10 respectively, an Unsure Action Knowledge Error occurs when a user chooses the immediate previously completed node action. For example, this error occurs if the user has just completed action node no. 1.1.2.4.1 Get Overhaul Equipment, and he chooses action node no. 1.1.2.4.1 Get Overhaul Equipment, rather than action node no. 1.1.2.4.2 Check Burnable Material Extinguished.

5. Transposition Error--In both the Action Tree Model and the Casualty Tree Model shown in Figures 3.9 and 3.10 respectively, a Transposition Error occurs when a user chooses a node action which is to the right of the correct choice node action or to the right of an ancestor of the correct action node. In this situation, a search is made for the largest and highest level not-yet-performed Action Tree Model node, which has been forgotten entirely by the user, or has been transposed with the user's choice node or one of its ancestor nodes. For example, in Figure 3.9, this error occurs if the previously completed action node was no. 1.2.5.3.2(not shown) and the user chooses no. 1.2.5.3.4 Dismiss Reflash Watch, then the

user has transposed the correct node no. 1.2.5.3.3 with node no. 1.2.5.3.4. The more interesting example is if the previously completed action node was no. 1.1.2.4 Verify Fire Out, and the user chooses node no. 1.2.5.1 Debrief the Fire Team. In this example the user has transposed the action node no. 1.2.4(not shown) with action node no. 1.2.5 Secure the Fire Team.

6. Incomplete Action Knowledge--In both the Action Tree Model and the Casualty Tree Model shown in Figures 3.9 and 3.10 respectively, an Incompleted Action Knowledge error occurs when the user doesn't re-choose the previously chosen action which wasn't properly completed. This error differs from the transposition error in that the user had previously chosen the correct action node, reflecting that his Knowledge Tree doesn't transpose that action node with other action nodes. This error case reflects the user's failure to realize that the action wasn't properly completed, as evident by the last "NOK" scene report as described in section III.B.2.a. For example, in Figure 3.9, this error occurs when the previously chosen action was action node no. 1.1.1.2.1 Locate Circuit Breakers, and a "NOK" report is received concerning this action, and then the user chooses action node no. 1.1.1.2.2 Open Circuit Breakers.

A student model must provide "a detailed analysis of each portion of the student's solution as necessary in order to diagnose errors" [Ref. 2: p. 9]. The Fire system's Student Model, using the Action Tree Model data structure and the current state of the game environment, fulfills this requirement which enables the Tutoring Module to provide the student user with the best appropriate guidance.

5. Tutoring Module

The tutoring module of ICAI systems must integrate knowledge about teaching methods, user explanation formats, and the knowledge domain of interest. A recent teaching strategy that has been successfully implemented on several systems is called "coaching." Coaching programs are not concerned with covering a predetermined lesson plan within a fixed time. Rather, the goal of coaching is to encourage skill acquisition and general problem-solving abilities by engaging the student in some activity like a computer game. In a coaching situation, the immediate aim of the student is to have fun, and skill acquisition is an indirect consequence. A successful computer coach must be able to

discern what skills or knowledge the student might acquire, based on his playing style, and to judge effective ways to intercede in the game and offer advice. The Fire system uses the coaching teaching strategy. [Ref. 3: p. 234]

The question of "when" a coach should interrupt the game is a complex issue and, in most ICAI coaching systems, depends on the knowledge domain being taught. Rather than allowing a student to perform more than one incorrect action without any feedback, most current systems including The Geometry Tutor [Ref. 6: p. 1], The LISP Programming Tutor [Ref. 2: p. 1], SPIRIT [Ref. 5: p. 1], and WEST [Ref. 3: p. 257], use the "immediate feedback" method [Ref. 2: p. 9]. This method is executed by keeping the tutoring module quietly in the background while the user inputs correct answers. If an incorrect input is diagnosed, then the tutoring module interrupts with advice.

The Fire system has adapted the immediate feedback interruption method primarily because a fire casualty scenario which allows more than one incorrect action, as in a real fire, would create an unpredictable course of events and result in confusing a student fire team leader. For example, given a fire casualty scenario status in which the fire has been extinguished and a reflash safety watch has been set. Then the fire team leader fails to order the compartment to be overhauled to verify no smoking embers exist, and fails to perform explosive gas tests. He then goes ahead and performs the next action of turning on blower fans to remove the smoke from the space. Now the scenario could change in numerous different ways, such as, the fan explodes due to explosive gases which may or may not cause personnel casualties or new fires. A different scenario is that the space is filled with fire due to the presence of hot embers and air flow from the fan. Therefore, to train the user what the correct order of recovery actions are, and

why they are performed in a specific order, immediate feedback is used to focus the user's attention on each of his specific actions in order to understand their sole impact if performed improperly.

The Fire system's Tutoring Module functions include correction of the user's misunderstandings and to provide the user an opportunity to take a time out from the game environment and learn about the action in which the user has exhibited a weakness. The user's misunderstandings, as detected by the Student Model, are addressed and approached using a different explanation strategy for each of the six error cases discussed in setion III.B.4. Additional tutoring information concerning the specific subactions or descriptions of the action missed is also available to the user prior to returning to the Game Environment. Using the Tutoring Module effectively, a user will not only learn why his answer is not the best choice of actions, but also be "steered" back in the direction of the correct answer.

a. Tutoring Strategies

By using the Student Model determined error case, the user's choice, and the last scene report, the Tutoring Module can effectively tutor the user on six different knowledge related concepts. Explanations developed for tutoring purposes were designed to be short, direct, easily understandable, and clearly displayed. The Tutoring Module is similar to SPIRIT [Ref. 5: p. 10], and the LISP Tutor [Ref. 2: p. 10], in that it will make the student user find the answer by himself. A short description of the six conceptual error cases' tutoring strategies addressed by this Tutoring Module follows. Their individual error case Student Model detection schemes are described in section III.B.4.

(1) Case 1: Wrong Task Error. The Wrong Task Error could result from the user not comprehending the casualty initiation information of the last scene report. It could also result from the user not "keeping the big picture" regarding the numerous actions that need to be acted upon and choosing randomly what to do next. This behavior, representing a user which has lost his ability to rank his many concerns, would lead to very unsafe conditions in a real fire.

The tutoring strategy, shown in Figure 3.20, used for this situation, is one that forces the user to refocus his attention on the immediate problem at hand. This is accomplished by politely informing him that his last action is not even relevant to what should be his highest priority, namely a personnel injury or equipment casualty, and then to "steer" him in the right direction by reminding him of the last scene report. This strategy closely parallels real fire situations where a good fire team member, who upon hearing the casualty related report from the scene and observeing the fire team leader order an action totally unrelated to the problem at hand, will quickly speak up and advise the leader that his action is incorrect and remind the leader of the last scene report.

User Name, your choice of:
Choose and activate a drain method type.

has nothing to do with the current most important action to be performed. The LAST REPORT:
The fire team is at the scene but the sound powered phones do not work.

should help you decide what to do next.

Figure 3.20 Wrong Task Tutoring Strategy

(2) Case 2: Weak Casualty Knowledge. The Weak Casualty Knowledge error case could result again from the user not comprehending the casualty initiation information of the last scene report or the user being weak in knowing the correct order the casualty recovery actions should be performed. This case appears only in the Complex Fire Casualty mode of operation which is discussed in section III.C.3. This user behavior generally represents a fire team leader with poor potential to control and recover from unexpected events, such as personnel, equipment, or fire reflash casualties.

The tutoring strategy, shown in Figure 3.21, used for this situation, is based on the philosophy that the potential casualty immediate recovery actions for personnel, equipment, or fire reflash casualties must be known by a fire team leader in a complex fire scenario. In recovering from a real shipboard fire, if these casualties occur, and are not effectively and quickly handled, a high probability of loss of life or increased equipment damage will result. Therefore, the user is given a very strong and direct report that he is taking the wrong action for the current casualty state. No hint of the correct action is given, which will force a poor fire team leader to continue to flounder, causing his casualty subtask score to reflect this weakness. A good leader, who knows his casualty recovery actions, will determine for himself the correct action and maintain a satisfactory casualty subtask score.

(3) Case 3: Forgetting Completed Action. The Forgetting Completed Action error case could result from a conservative user which has difficulty in keeping "the big picture," and when in doubt as to whether he has already completed some action, he reorders its completion. This case could also result from the user not thoroughly understanding the subaction of a specific action. This

User Name; you have FAILED TO REALIZE that a REFLASH, PERSONNEL INJURY or EQUIPMENT FAILURE CASUALTY has just occurred or you've taken the WRONG IMMEDIATE ACTION!

You MUST take the CORRECT IMMEDIATE ACTIONS in the CORRECT ORDER to CONTROL and RECOVER from this CASUALTY.

Figure 3.21 Weak Casualty Knowledge Tutoring Strategy

could lead him to thinking an action hasn't been completely performed when in fact it has. This error, in a real shipboard fire, can only indirectly lead to a more dangerous situation as a result of the time wasted to recomplete an action. During this time, a fire could possibly grow in intensity or spread to adjacent spaces. This situation of wasting time recompleting an already completed action would rarely occur. Generally a good fire team member, usually the fire team leader's phone talker, reminds the leader that his ordered action had already been completed.

The tutoring strategy, shown in Figure 3.22, used in this situation, is to inform the user that he has already completed a specific action. He will then be shown the subactions of the task which he thought wasn't completed to give him a thorough understanding of that task. If the user desires, he can learn more about these subactions by asking the Tutoring Module for further training of this specific action using the Action Node Breakdown feature described in section III.B.5.b. He will then have to determine for himself what is the next best action to perform.

(4) Case 4: Unsure Action Knowledge. The Unsure Action Knowledge error case results from a user not comprehending the last scene report's implication that the previously ordered action has been completed. This case

User Name, YOU HAVE ALREADY COMPLETED the action:
Get to the scene equipped and ready to go.

STUDY AND LEARN the following information to help you next time you have a similar situation. Below is a breakdown of the specific subactions, which make up the action: Get to the scene equipped and ready to go.

1. Get the fire team dressed.
2. Get the fire team to the scene.
3. Take charge of the scene.
4. Establish phone communications with the Repair Locker.
5. Charge the fire hoses.

Figure 3.22 Forgetting Completed Action Tutoring Strategy

could also result from a user not understanding thoroughly the subactions of the previous action. This would be evident of a user who had received reports that each subaction had been completed, but still didn't think the action was now considered complete. As in Case 3, this could only indirectly lead to a more dangerous fire casualty as a result of wasting valuable time recompleting this action.

The tutoring strategy, shown in Figure 3.23, is to force the user to refocus his attention on the last scene report and inform him that the report implied the previous action was complete. He will then be shown the subactions of the previous action, which he thought wasn't completed, to give him a thorough understanding of that task. If the user desires, he can learn more about these subactions by asking the Tutoring Module for further training of this previous action's subactions using the Action Node Breakdown feature described in section 3.B.5.b. The user will then have to determine for himself what is the next best action to perform.

User Name, the LAST REPORT:
The Explosive Gas tests indicate negative explosive gas.

SHOULD HAVE indicated to you that the action:
Perform explosive gas tests.

has JUST BEEN COMPLETED.

STUDY AND LEARN the following information to help you next time you have a similar situation. Below is a breakdown of the specific subactions, which make up the action: Perform explosive gas tests.

1. Get explosive gas tester.
2. Have the explosive gas tester tested.
3. Sample the space, which had the fire, for explosive gases.

Figure 3.23 Unsure Action Knowledge Tutoring Strategy

(5) Case 5: Transposition Error. The Transposition Error case results from a user not understanding the correct order which the recovery actions are to be performed, or either forgets, or never had the knowledge, of the correct action or its higher level governing action, which should now be performed. Both of these causes are signs of a potentially poor fire team leader who currently is not capable of effectively directing the fire team. He also doesn't understand the reasons for executing the recovery actions in a prescribed order or the potentially bad consequences which could result from not performing the correct action at the proper time.

The tutoring strategy, shown in Figure 3.24, is to focus on why the user's choice of actions or the higher level concept action which his action belongs, is incorrect. This is accomplished by describing a potentially bad consequence which could occur as a result of not performing the immediate predecessor's highest level missed action. The user is first reminded of his action or its higher level conceptual action which he has chosen. Then

the user is informed that he must first perform the immediate predecessor action to this action. This immediate predecessor may or may not be the actual correct answer. This approach is to "steer" the user back in the direction of the correct action. If a user has chosen an action which is very distant from the previous correct action, this tutoring strategy will provide an explanation using very high level concepts. This enables the user to understand more easily his conceptual error. For example, if a user should have chosen the action, "Set Fire Boundaries.", but instead chose to "Secure the Fire Team.", the Tutoring Module would find the highest level ancestor node action of these actions to "steer" the user back. The explanation would inform the user that before he could "Cleanup After the Fire," he first must "Put the Fire Out."

To teach the user the potential impact of forgetting or transposing his choice's immediate predecessor action and his chosen action, a bad consequence explanation is presented. This bad consequence addresses these actions, which were apparently transposed, and attempts to clearly emphasize and make a lasting impression on the user why these action should never be transposed.

The bad consequence explanation predicate, shown in Figure 3.25, consists of two sequential action node numbers located on the same level of the Action Tree Model, and the corresponding textual description of the bad consequence. If a bad consequence description was required for every node with every other node in the Action Tree Model, then for N nodes, N-Squared bad consequence descriptions would be required. This is not desirable for many reasons. A user is only tested on a local area of the Action Tree Model for each situation. This area generally includes the correct answer's immediate predecessors and successors. Also, it doesn't make sense for subaction nodes

User Name, BEFORE you can perform the action:
Set fire boundaries.

YOU MUST FIRST perform the action:
Deenergize the space.

OTHERWISE
if the fire's source is electrical, it will continue to help the fire grow in intensity and if the fire's source wasn't electrical, electrical cabling in the fire space will soon breakdown due to the heat and cause new fires.

STUDY AND LEARN the following information to help you next time you have a similar situation. Below is a breakdown of the specific subactions, which make up the action: Deenergize the space.

1. Have Repair Locker locate on the ship's electrical drawings all circuit breakers which isolate electrical power to the space on fire.
2. Send electrician to locate and open all circuit breakers which isolate electrical power to the space on fire.

Figure 3.24 Transposition Error Tutoring Strategy

to be related to their ancestors or descendents using bad consequence descriptions. Finally, the excessive memory requirements to store these additional descriptions and the resulting slower Game Environment response time would detract from the Tutoring Module's training effectiveness.

An upper bound for the number of bad consequence descriptions required for the Action Tree Model structure can easily be determined. A bad consequence description is required for every action node and its immediate predecessor node, if one exists. In Figure 3.2, the left most nodes on each level do not have predecessors. The number of left most nodes is equal to the number, L, of levels. If the Action Tree Model consisted of N nodes and all nodes were expandable nodes except the bottom level nodes, then the number of bad consequence descriptions required would be N-L. But in Figure 3.9, the Action Tree Model has test procedure and non-expandable nodes on levels

other than the bottom level. This results in some internal tree nodes not having immediate predecessors and consequently no required bad consequence descriptions. Since the location of these nodes is determined by the specific actions which make up the Action Tree Model, the exact number of bad consequences which aren't required cannot be exactly determined for any general action tree model. Therefore, an Action Tree Model, with N nodes and L levels, requires at most N-L bad consequence descriptions. Fire's Action Tree Model has 100 nodes, 8 levels, and 89 bad consequences.

Following the bad consequence explanation, the user is shown a description or the subactions of the immediate predecessor action to his chosen action. Since he may have forgotten this action or has no knowledge of it, this display will give him time to thoroughly learn about it. If the user desires, he can learn more about these subactions by asking the Tutoring Module for further training on these subactions using the Action Node Breakdown feature described in section III.B.5.b.

(6) Case 6: Incomplete Action Knowledge. The Incomplete Action Knowledge error case results from a user not understanding all of the subactions of a given action. This is evident when a user prematurely chooses the next sequential ordered action when the previous action hasn't been completed. The tutoring strategy, shown in Figure 3.26, used for this situation, forces the user to focus his attention on the last scene report and informs him that the report implied the previous action hasn't been completed. He is then shown the subaction of the previous action which he thought was completed to give him a thorough understanding of that action. If the user desires, he can learn more about the subactions by asking the Tutoring Module for further training on these subactions using the

```
node_action([1,1,1,1,2,1,2],
['Identify the deck, frame, center line relationship,',
 'and function of the space.']).
node_action([1,1,1,1,2,1,3],
['Have Repair Locker locate on the ship's ',
 'compartment drawings the space with the fire.']).

bad_consequence([1,1,1,1,2,1,2],[1,1,1,1,2,1,3],
['a lot of time will be wasted trying to randomly',
 'locate a space number on any of a number of ',
 'charts.  Using the deck, frames, center line ',
 'relationship numbers the correct chart and space',
 'can quickly be located.']).


node_action([1,1,2,3],['Set a Reflash Watch.']).
node_action([1,1,2,4],['Verify all fires are out.']).

bad_consequence([1,1,2,3],[1,1,2,4],
['if the fire reflashes the overhaul team could be ',
 'trapped in the space, resulting in possible ',
 'personnel injury with no fire extinguishing ',
 'equipment present.']).
```

Figure 3.25 Examples of Bad Consequence Explanation Predicates

Action Node Breakdown feature described in section III.B.5.b. He will then determine for himself what is the next best action.

b. Action Node Breakdown Feature

The second function of the Tutoring Module discussed in section III.B.5, allows the user an opportunity to take a "time out" from the Fire system Game Environment, and pursue additional tutoring on the description or subactions of a particular action which the user demonstrated a weakness. Using the flexible structure of the Action Tree Model, and the easily manipulative node-numbering system, any particular action node can easily be located along with its subaction nodes. Consequently, the subaction nodes can also be described in the same manner until leaf action nodes are encountered. This feature, as shown in Figures 3.27, 3.28, and 3.29, allows the user to

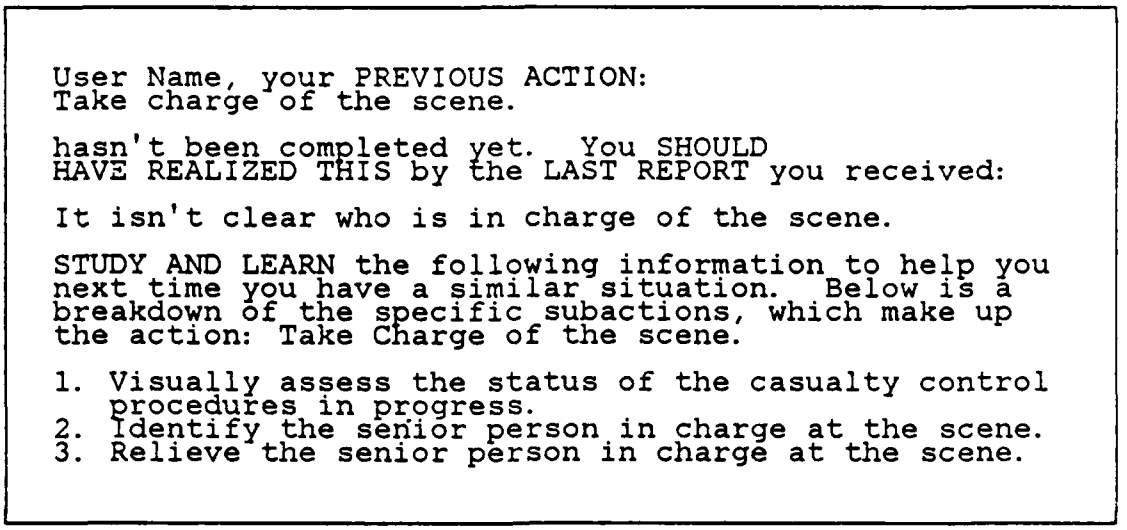


Figure 3.26 Incomplete Action Knowledge Tutoring Strategy

designate which subaction he desires further breakdowns on, inorder to reveal their subactions or descriptions. When the user has completed his review of an action's subactions, he is

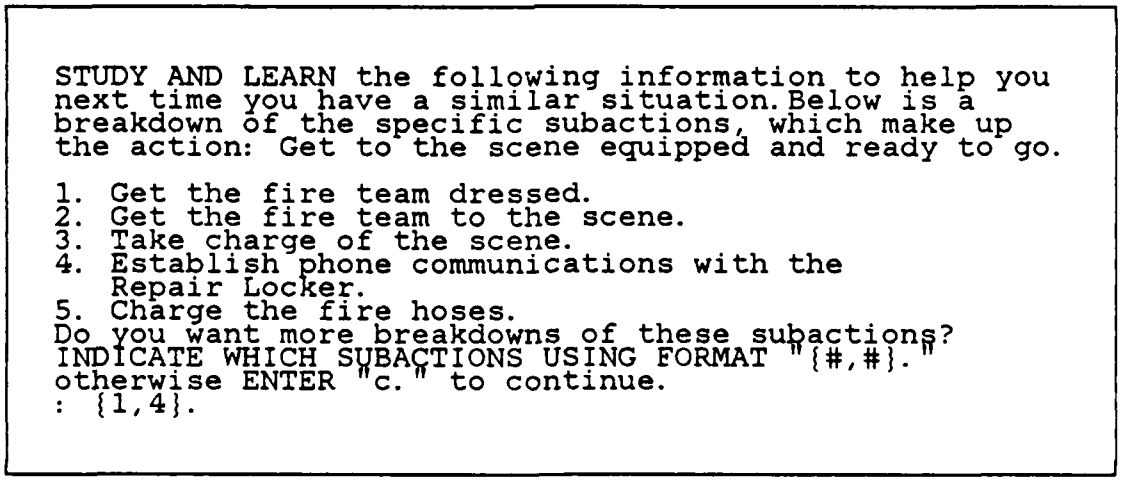


Figure 3.27 Action Node Breakdown Feature Display One

returned to the Game Environment and presented with the exact same game state situation, as shown in Figure 3.30, which he just made an error on. This allows him an

```
     ***  REMINDER OF PREVIOUS ACTION BREAKDOWN  ***
Below is a breakdown of the specific subactions, which
make up the action:
Get to the scene equipped and ready to go.

1. Get the fire team dressed.
2. Get the fire team to the scene.
3. Take charge of the scene.
4. Establish phone communications with the
   Repair Locker.
5. Charge the fire hoses.

    ***  NEW SUBACTION BREAKDOWN  ***
Below is a breakdown of the specific subactions,
which make up the action:
Get the fire team dressed.

1. Dress the fire team in Battle Dress.
2. Dress the fire team in OBA's.
3. Equip the fire team with fire fighting equipment.

Do you want more breakdowns of these subactions?
INDICATE WHICH SUBACTIONS USING FORMAT "{#,#}."
otherwise ENTER "c." to continue.
```

Figure 3.28 Action Node Breakdown Feature Display Two

opportunity to immediatedly correct his last knowledge error as a result of what he learned during his "trip" to the Tutoring Module.

C. SPECIAL SYSTEM MODES OF OPERATION

1. Individual Equipment Operations and Basic Damage Control Actions

This mode of operation allows a user an opportunity to learn about new basic tasks or concentrate on relearning basic tasks which have given him trouble. The user can choose from a list of 25 tasks which include individual equipment operations or basic damage control actions. He can choose the tasks in any order he prefers to learn them. Specific tasks will be displayed with the reference which the specific task knowledge test is taken. All of these tasks are Test Procedure Node Actions as described in section III.B.1.a.(3), and executed as described in section III.B.2.b. Following the completion of the user's tasks,

```
     ***  REMINDER OF PREVIOUS ACTION BREAKDOWN  ***
Below is a breakdown of the specific subactions, which
make up the action:
Get to the scene equipped and ready to go.

1. Get the fire team dressed.
2. Get the fire team to the scene.
3. Take charge of the scene.
4. Establish phone communications with the
   Repair Locker.
5. Charge the fire hoses.

     ***  NEW SUBACTION BREAKDOWN  ***
Below is a breakdown of the specific subactions,
which make up the action:
Establish phone communications with the Repair Locker.

1. Hold the set of phones in your left hand.
2. Unhook the right side of the neck strap from the
   breastplate, put the strap around your neck, and
   then fasten it to the breastplate again.
3. Put the earphones on and adjust the headband so
   that the center of the ear piece is directly over
   the opening into the ear.
4. Insert the plug into the jack box and screw the
   collar on firmly.
5. Adjust the mouthpiece to bring it directly in front
   of your mouth when you stand erect.
6. Test the phones with someone on the circuit.

Do you want more breakdowns of these subactions?
INDICATE WHICH SUBACTIONS USING FORMAT "{#,#}."
otherwise ENTER "c." to continue.
```

Figure 3.29 Action Node Breakdown Feature Display Three

the Fire system will exit, requiring restarting of the program for further training.

2. Individual Complex Damage Control Actions

This mode of operation allows a user an opportunity to learn new complex tasks or concentrate on relearning complex tasks which have given him trouble. Again, he can choose in any order, up to 25 different tasks. All of these tasks are Expandable Node Actions, as described in section III.B.1.a.(1), and executed as described in section III.B.2.b. The individual tasks are presented exactly as they would be in the Complex Fire Casualty mode. Upon completion of each task, the user will be given his

```
User Name, now that you understand why your last
choice of actions wasn't the best choice, I'll repeat
the last report you received and the same choices.
CHOOSE THE BEST ACTION. GOOD LUCK!

                                    User Name's Score Level
                                          1 of 3    33%
User Name, you are the "FIRE TEAM LEADER."

YOUR LAST REPORT WAS:
The fire team is at the scene but the sound powered
phones do not work.

FIRE TEAM LEADER, indicate your most important,
                  or next action to be completed.
1. Deenergize the space.
2. Have fire team member recheck proper operation of
   malfunctioning equipment.
3. If equipment is still malfunctioning, check local
   area for replacement.
4. Choose and activate a drain method type.
5. Have a second team member check for proper
   operation of the equipment.
6. Obtain a satisfactory first Oxygen Test.

7. HELP
8. QUIT
: 2.
```

Figure 3.30 Post Tutoring Module Trip Display

performance score for that task. All incorrect answers will be addressed by the Fire system's Tutoring Module as described in section III.B.5. One exception, is when a casualty task is presented, and actions are chosen out of order, a detailed explanation of why the user's choice was incorrect will be given. Following the completion of the user's tasks, the Fire system will exit, requiring restarting of the program for further training.

3. Complex Fire Casualty

This mode of operation is the primary method of using the Fire system. It places the user in the role of a shipboard Fire Team Leader directing a fire fighting team in a simulated fire casualty environment. This mode is executed using all functions and features described in section III.B. The simulated main fire casualty scenario

begins with a shipwide announcement of the fire's type and location and ends when the fire team and reflash watch have been secured. Following the completion of the Performance Report display, as discussed in section III.B.2.d, the user will have an opportunity to "QUIT" or choose the mode of operation described in the next section.

4. Previous Fire Casualty Operation Training

This mode of operation allows the user to be retested on the tasks which he just performed in the Complex Fire Casualty mode. The tasks are sorted with the best performance first, worst next, second best third, second worst fourth, and so on. This order keeps the user's interest and motivation high, even during tasks which he didn't do well in before. These individual tasks are presented in a similar manner as they were in the Complex Fire Casualty mode. Upon completion of each task, the user will be able to see his score for that task. Following the completion of the previous tasks, the Fire system will exit, requiring restarting the program for further training.

IV. USING THE FIRE SYSTEM ON US NAVY SHIPS

A. MEMORY REQUIREMENTS

To analyze the feasibility of using the Fire system on a microcomputer aboard US Naval ships, the memory requirements for the system's files must not be excessive. The Fire system's files were transferred from a VAX 11/780 Unix file system to a standard 5.25 inch, one sided, double density floppy disc, via a Tandy 1000 personal computer. The main program and its seven major supporting files consume 128K bytes of memory. The other 54 supporting node action files consumed 72K bytes of memory. Therefore, the Fire system's memory requirements were 200K bytes of a possible 360K bytes available on this floppy disc. The Prolog-86 Interpretter [Ref. 12: p. 1], required to execute the Fire system on a microcomputer requires approximately 100K bytes of memory. Therefore, the Fire system and the required Prolog-86 Interpretter can easily fit on one floppy disc which is very convenient in an on board microcomputer training environment.

B. PROGRAM LANGUAGE REQUIREMENT

The Fire system was developed on a VAX 11/780 Unix System. This system uses the Prolog language as described in Cocksin and Mellish [Ref. 13: p.1]. The Clocksin and Mellish Prolog, with a few minor code changes, can be converted to meet Prolog-86 code requirements, a microcomputer version of Prolog.

C. EXPANDING THE FIRE SYSTEM FOR DIFFERENT KNOWLEDGE DOMAINS

The Fire system, as shown in Figure 3.1, consists of four modes of operation, four special modules, and an Action Tree Model. The Action Tree Model contains almost all of

the Fire Team Leader's expert level domain knowledge. Casualty team leaders for fire teams, flooding teams, medical emergency teams, and numerous other casualty teams required on US Naval ships, have very similar control or strategy knowledge. They all act on reports from the immediate casualty scene to determine their course of action. The Fire system structure can be converted to train these other types of casualty team leaders by swapping out the Fire Team Leader Action Tree Model for another casualty leader's Action Tree Model. This ICAI tutoring concept, would be very valuable in a shipboard training environment where the main Fire system could be loaded into a microcomputer followed by any one of many casualty team leaders' Action Tree Models which the user requires training.

V. SUMMARY

A. DISCUSSION OF THE RESEARCH QUESTIONS

This study was undertaken to develop a solution to the problem of keeping stable, trained, and fully qualified inport fire team organizations on US Naval ships. Intelligent Computer Aided Instructional(ICAI) tutoring systems were proposed to significantly reduce the training time while increasing the quality of the individual training for damage control personnel. To investigate this proposal, a Fire Team Leader ICAI Tutoring System was developed. Its design and implementation issues were described in Chaper III. The written references for this ICAI system's fire team leader's knowledge domain development were common damage control references found on all US Naval ships. The feasibility of actually using this Fire Team Leader Learning Center System, or Fire, was investigated in Chaper IV.

Chapter IV demonstrated that it is feasible to use an ICAI tutoring system, such as Fire, on a microcomputer on board every US Naval ship. The only material's required are a microcomputer, a copy of the Prolog 86 Interpreter, the Fire system program and supporting files on floppy discs, and a Megabyte of memory. Chapter III's description of the functions and features of the Fire system provide a challenging, realistic training environment for fire team leaders to combat computer simulated fires. Therefore, the answer to the first research question this study asks is "yes", it is feasible to develop an effective, challenging, expert, fire fighting system program, for the purpose of training fire team leaders in combatting fires on US Naval Ships.

ICAI tutoring systems, which tutor a student towards an expert level of knowledge in an area such as fire fighting,

require both strategic, or control knowledge, along with factual knowledge. The strategic knowledge is instrumental in determining the use of factual knowledge for different scenario situations. US Navy damage control references, used to aid in the development of the Fire Team Leader Learning Center, were good in the factual knowledge of how to operate equipment or how to perform certain damage control related actions. But they were poor in describing the strategic knowledge which fire team leaders should have and expert behaving programs require. This information tells why one action is performed rather than another action for a given situation. This "why information" gives each action a reason for existing in the Action Model Tree, in that, by performing it correctly, at the right time, prevents a potentially bad consequence which otherwise might occur. The "why information" implemented in this Fire system was based on the author's previous shipboard drill training environment experience and partially on damage control reference manuals. Therefore, the answer to the second research question this thesis addresses is that a rule based expert system, which will correctly analyze casualty symptoms and make proper decisions to extinguish fires, can only be partially derived from US Navy Damage Control Training references.

B. ICAI TUTORING SYSTEM LIMITATIONS

1. Realistic Simulation

The success of an ICAI Tutoring system such as Fire, relies heavily on a realistic simulation of a shipboard fire. Fire's simulation of a fire is implemented using textual report descriptions which originate at a hypothetical fire casualty scene. This approach was justified by stating that an actual fire team leader's location on a ship is not at the compartment of the fire, and therefore he only understands the fire's status from the

reports he receives from the fire scene. This simulation still only partially reflects a real shipboard casualty environment. A real fire team leader also has the sense of sight, allowing him to visually see which equipment is present, resulting in no unnecessary orders such as "Get the Oxygen Tester". Using his sense of smell, the real fire team leader can sometimes detect when smoke is spreading to adjacent spaces. Finally, using his sense of feel, the fire team leader can feel the heat generated by the fire, which based on its changing intensity, would reflect if the fire is increasing or decreasing in size. The fire team leader, who wears an oxygen breathing apparatus and protective fire fighting clothing, will often times not be able to use his sense of smell or feel.

In order to incorporate the sense of sight, Fire could be incorporated onto a graphics capable computer or be accompanied by a detailed picture book for different fire scenarios. In addition to the scene reports which Fire displays, a graphical presentation displaying the equipment, smoke, people and other objects which a Fire Team Leader can see could be displayed. This system would provide a more realistic simulation, more entertaining ICAI tutoring system, and a more effective training environment then the basic Fire system described in this thesis.

2. Action Tree Model Accuracy

The knowledge which can be learned from Fire is only as good as the development of the Action Tree Model. The primary question of whether the US Navy's Fire Fighting Policies can be properly implemented in this structure is the key to the accuracy of the Action Tree Model. The development of an Action Tree Model to accurately reflect how to combat a fire on a US Navy ship is difficult because:

1. Every fire is different and sometimes there are more than one correct sequence of steps in combatting the same fire.

2. US Navy Damage Control references focus, in most cases, on the basics of fire fighting using simple examples. These references provide very little assistance in determining which actions are more important, or must be performed prior to other actions. The reasons, or potentially bad consequences relating to the importance of one action over another action are seldom given. This lack of information results in the Fire system's programmer having to make decisions which may not correspond to the existing Navy's policies.

3. To prevent the tutoring system knowledge from becoming out dated, the Action Tree Model must be adaptable to frequently changing Navy policies regarding fire fighting procedures or equipment requirement changes. Fire's Action Tree Model is adaptable to these changes as described in section III.B.1.c.

3. Knowledge Base Structure

The Fire system's knowledge base required structure needs to represent all possible states which a fire casualty could be in. Traversing this structure from a given start state, such as "announce the fire casualty", to a goal state, such as "the fire is out and the ship restored to normal operation", a path of actions could be followed by a ICAI system to tutor students. The first structure for the Action Tree Model which was chosen was a means-ends analysis approach as described in [Ref. 14: p.146-156]. This structure used a difference-procedure table which allowed procedures or next actions to be chosen which would reduce the difference between the current state and the goal state. Each next action chosen, contained preconditions or other actions which had to be completed prior to using the desired action. The Means-Ends Analysis method works very well for problems where many actions have the same preconditions so that numerous paths could be evaluated and the shortest path, or least number of actions, could be found to arrive at the goal state.

In the fire casualty problem-solving domain, the recovery actions which must be used to traverse from the start state to the goal state are all unrelated actions. These actions are unrelated in respect to having no common preconditions which would allow multiple paths to the goal

state. The recovery actions do have a preferred order of execution. This results in every recovery action becoming a direct precondition of its successor action. For example, the precondition of "Perform the Second Oxygen Test" is "Perform the First Oxygen Test". The precondition of "Perform the First Oxygen Test" is "Test the Oxygen Test Equipment". Applying the means-ends-analysis approach to this problem domain results in one long sequential list of actions to be performed to reach the goal state. This approach was not chosen for Fire's Action Tree Model. The resulting simple sequential list of actions generated by this approach were contained in a complex difference-procedure table which executes numerous recursive calls. This would have resulted in a slower response ICAI tutoring system. It was also a complex issue to teach the subactions of a particular action to a student while the system was executing the difference-procedure table.

The Action Tree Model structure which was chosen is a multi-level top-down designed tree structure. This structure worked well for actions which contained distinct subactions which are performed in a preferred order. These subactions could also be further broken down into their subparts. For these types of actions, which had many levels of subdivisions, this structure was very effective in teaching complex actions by learning the numerous basic subactions which make up the complex action. Students which have difficulty in learning the subactions of a particular action, can be tutored using higher level, more general conceptual descriptions of actions which also contain those subactions. For actions which do not contain distinct ordered subactions, it is unclear if the action should stand alone, such as Fire's non-expandable node actions or whether some form of amplifying information should be presented which could effectively teach the action.

The Fire system used both order-dependent and informational test procedure nodes to reflect the nodes which contained amplifying information. Type classifying the action nodes is based on the amount of knowledge the designer has for each node action. If the designer of the Action Tree Model cannot find sufficient knowledge about a particular action from a reference book or a human expert, the action node will probably be classified non-expandable and not taught very well to the user. If some information can be found about an action, but not enough to describe exactly what the subactions are and the reasons for the order they're performed, then the action node will be classified a test procedure node. In some test procedure nodes, amplifying information such as safety precautions or guidelines for performing the action can confuse rather then teach a student. If everything is known about an action, including all of its subactions and the reasons for their order of execution, then the action node is classified an expandable node and a user can be taught everything about that action.

The design of an Action Tree Model should strive to contain expandable action nodes on all but the last level of its tree structure. This cannot always be accomplished. Actions cannot always be broken down into the same number of subdivisions. In summary, Fire's Action Tree Model structure was chosen because it was easy to traverse from an initial start state to a goal state along its fourth level, and also allowed an easy method to teach the student user the subactions of higher-level actions.

4. Careless Errors

Most ICAI tutoring systems address the issue of dealing with the "careless error". The careless error occurs when a user chooses an answer without using any thought process. For example, the user doesn't read the

action menus displayed in Fire and enters a "1." each time. The Fire system doesn't utilize dummy menu items for detecting careless errors such as the action "Go Fishing". The action menus often contain fill in action items, as described in section III.B.2.c, which if chosen, will result in one line responses such as "You've already completed that action.", or "That action has nothing to do with the current most important action which should be performed". Generally careless users will receive these short tutoring reports. The careless user's scores suffer from careless errors.

C. BENEFITS OF RESEARCH

This study investigated the concept of using ICAI tutoring systems for administering individual training for US Naval personnel in damage control fire fighting principles. This research can also result in benefits in other areas of shipboard training.

1. In this period of time when microcomputers are reasonably priced, the US Navy could use these computers for effective training of personnel using Navy tailored programs.
2. By using ICAI programs to train personnel, uncountable man hours in lecturing and teaching can be saved by supervisors.
3. Effective damage control casualty training can be conducted on computers at times when a ship's schedule can't support actual casualty drills.
4. This program, which conducts training on fires, is only one of many types of casualty training which is conducted on ships. As demonstrated by this program, other computer programs could be developed for casualty training for engineering, weapons, nuclear power, medical emergencies, and almost all other technical areas where casualties could occur.
5. The Naval Education and Training Command could acquire and make available ICAI programs to aid in the Navy's personnel Qualification Standard(PQS) Program which requires extensive knowledge in the operations and casualty control of each watchstation.

APPENDIX A

DEMONSTRATION OF COMPLEX FIRE SIMULATION

Steve, please enter the time of day(0001-2400): 1000.

Thank You. Steve, if you are not familiar with my operation, request
HELP, otherwise choose an area of Operations and/or Casualty
Control that you'd like to challenge me in!
1. HELP
2. Individual Equipment Operations and Basic Damage Control Actions
3. Individual Complex Damage Control Actions
4. Complex Fire Simulation

Enter a 1., 2., 3., or 4. for desired category: 4.

Steve, please enter your ability level as follows:
1. Beginner. 2. Advanced. 3. Expert. for this category.
Enter a 1., 2., or 3. for level:
: 2.
GOOD LUCK!

next screen display

DING DING DING DING DING
FIRE FIRE FIRE FIRE
FIRE IN ENGINEERING STOREROOM . COMPARTMENT 1 - 22 - 1 - A

AWAY THE REPAIR PARTY FIRE TEAM AWAY!

Steve , you are now the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Extinguish all the fires.
2. Set a Reflash Watch.
3. Set fire boundaries.
4. Approach the fire.
5. Deenergize the space.
6. Get to the scene equipped and ready to go.

7. HELP
8. QUIT
: 1.

next screen display

Steve. BEFORE you can perform the action:
Remove all fires.

YOU MUST FIRST perform the action:
Isolate the fire.

OTHERWISE
the time and men and equipment you spend trying to remove the fires in one location will be wasted because the fire will continue to spread to new spaces until it is isolated.

(Enter a "c." to continue.)c.

next screen display

STUDY AND LEARN the following information to help you next time you have a similar situation.
Below is a breakdown of the specific sub-actions. which make up the action:
Isolate the fire.

1. Get to the scene equipped and ready to go.
2. Deenergize the space.
3. Set fire boundaries.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "#,#."
otherwise ENTER "c." to continue.
: 1,3.

next screen display

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions. which make up the action:
Isolate the fire.

1. Get to the scene equipped and ready to go.
2. Deenergize the space.
3. Set fire boundaries.

*** NEW SUB ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions. which make up the action:
Get to the scene equipped and ready to go.

1. Get the fire team dressed.
2. Get the fire team to the scene.
3. Take charge of the scene.
4. Establish phone communications with the Repair Locker.
5. Charge the fire hoses.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT " #.= ."
otherwise ENTER "c." to continue.
: 1.3.

next screen display

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions, which make up the action:
Get to the scene equipped and ready to go.

1. Get the fire team dressed.
2. Get the fire team to the scene.
3. Take charge of the scene.
4. Establish phone communications with
   the Repair Locker.
5. Charge the fire hoses.

*** NEW SUB ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions, which make up the action:
Get the fire team dressed.

1. Dress the fire team in Battle Dress.
2. Dress the fire team in OBA's.
3. Equip the fire team with fire fighting equipment.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "#,#."
otherwise ENTER "c." to continue.
: 1,2.

next screen display

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions. which make up the action:
Get the fire team dressed.

1. Dress the fire team in Battle Dress.
2. Dress the fire team in OBA's.
3. Equip the fire team with fire fighting equipment.

*** NEW SUB ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions. which make up the action:
Dress the fire team in Battle Dress.

1. Get personnel into long sleeve shirts.
2. Get personnel to button up shirts at the
   neck and wrists.
3. Get personnel to place their pants legs
   into their stockings.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT " =,= ."
otherwise ENTER "c." to continue.
: c.

Enter "c." to continue: c.

[next screen display]

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions, which make up the action:
Get the fire team dressed.

1. Dress the fire team in Battle Dress.
2. Dress the fire team in OBA's.
3. Equip the fire team with fire fighting equipment.

*** NEW SUB ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions, which make up the action:
Dress the fire team in OBA's.

1. Have personnel get OBA's out of the
   Repair Locker.
2. Have personnel check their OBA hoses
   for proper connections.
3. Have personnel strap their OBA's
   firmly onto their bodies.
4. Have personnel install their OBA cannisters.
5. Have personnel don their OBA face pieces.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "[#,#]."
otherwise ENTER "c." to continue.
: 4,5.

[next screen display]

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***
Below is a breakdown of the specific sub-actions, which make up the action:
Dress the fire team in OBA's.

1. Have personnel get OBA's out of the
   Repair Locker.
2. Have personnel check their OBA hoses
   for proper connections.
3. Have personnel strap their OBA's
   firmly onto their bodies.
4. Have personnel install their OBA cannisters.
5. Have personnel don their OBA face pieces.

next screen display

*** NEW SUB ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions. which make up the action: Have personnel install their OBA cannisters.

1. Remove tear-off cap of canister by pulling tab backward and downward. exposing copper foil seal. Discard cap.

WARNING

When removing the candle cover. the lanyard must not be pulled so that the cotter pin is removed. Removal of the cotter pin fires the candle and starts generating oxygen. If this happens while the copper foil is intact. internal pressure in the cannister will build up causing the copper foil or cannister seam to rupture.

2. Remove the canister candle cover by rotating swivel plate 180 degrees. and push it down toward center of the canister, leaving cover dangling by the lanyard. Do not pull lanyard at this time.
3. Insert the canister(with neck up and concave, or ribbed, side toward body) upward into the guard and breastplate assembly until the canister is firmly retained by the bail. The canister is now locked in standby position. with the copper foil seal still intact.

Enter a "c." to continue.

next screen display

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action: Dress the fire team in OBA's.

1. Have personnel get OBA's out of the Repair Locker.
2. Have personnel check their OBA hoses for proper connections.
3. Have personnel strap their OBA's firmly onto their bodies.
4. Have personnel install their OBA cannisters.
5. Have personnel don their OBA face pieces.

Enter a "c." to continue.

next screen display

*** NEW SUB ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions. which make up the action: Have personnel don their OBA face pieces.

1. Place head harness straps over front of face piece.
2. Insert chin into chin stop of the face piece.

removed user session lines

Enter a "c." to continue.

[next screen display]

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Get to the scene equipped and ready to go.

[removed user session lines]

*** NEW SUB ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Take charge of the scene.

1. Visually assess the status of the casualty
   control procedures in progress.
2. Identify the senior person in charge at the scene.
3. Relieve the senior person in charge at the scene.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "[#,#]."
otherwise ENTER "c." to continue.
[: [3].

[next screen display]

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Take charge of the scene.

1. Visually assess the status of the casualty
   control procedures in progress.
2. Identify the senior person in charge at the scene.
3. Relieve the senior person in charge at the scene.

*** NEW SUB ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Relieve the senior person in charge at the scene.

1. When ready to receive briefing from the man
   in charge state to him.
   "I am ready to relieve you."
2. Receive briefing from the man in charge.
   (Obtain who, what, when, where, and how
   information about the casualty if available.)
3. When ready to take charge of the scene state,
   "I relieve you."
4. Announce to all personnel at the scene that
   you are the man in charge.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "[#,#]."
otherwise ENTER "c." to continue.
: c.

Enter "c." to continue: c.

[next screen display]

*** REMINDER OF PREVIOUS ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Isolate the fire.

1. Get to the scene equipped and ready to go.
2. Deenergize the space.
3. Set fire boundaries.

*** NEW SUB ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Set fire boundaries.

1. Have Repair Locker locate on ship's compartment
   drawings all adjacent spaces and all physical
   openings to the space including doors,
   ventilation ducts, and drains.
2. Send fire boundary teams to shut all physical
   openings to the space with the fire.
3. Send fire boundary teams to each adjacent
   space with portable fire equipment in order
   to verify the boundaries are containing the fire.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "[#,#]."
otherwise ENTER "c." to continue.
: c.

[removed user session lines]

next screen display

Steve 's Score Level
0 of 1 0 %
Steve , you are the " FIRE TEAM LEADER ."
[removed user session lines]

FIRE TEAM LEADER, you MUST now DIRECT THE ACTIONS to complete the following TASK: Deenergize the space.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Store all equipment away in the Repair Locker.
2. Lead the fire team to the Repair Locker.
3. Choose and activate a drain method type.
4. Have Repair Locker locate on the ship's electrical drawings all circuit breakers which isolate electrical power to the space on fire.
5. Obtain a satisfactory second Oxygen Test.
6. Send electrician to locate and open all circuit breakers which isolate electrical power to the space on fire.

7. HELP
8. QUIT
: 2.

[next screen display]

Steve, BEFORE you can perform the action:
Put away the fire fighting equipment.

YOU MUST FIRST perform the action:
Dewater the space.

OTHERWISE
personnel will not have the equipment to properly dewater the space
which may result in increased equipment water damage.

(Enter a "c." to continue.)c.

[next screen display]

STUDY AND LEARN the following information to help you next time you have a similar situation.
Below is a breakdown of the specific sub-actions, which make up the action:
Dewater the space.

1. Determine the amount of water in the space.
2. Choose and activate a drain method type.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "#.#."
otherwise ENTER "c." to continue.
: c.

[removed user session lines]

next screen display]
Steve 's Score Level
5 of 7 71 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Send electrician to locate and open all circuit
breakers which isolate electrical power to the
space on fire.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The electrician has open all circuit breakers
which provide power to the space.

FIRE TEAM LEADER, you receive the following report:
The action: Deenergize the space.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The space has not yet been deenergized.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Get to the scene equipped and ready to go.
2. Extinguish all the fires.
3. Approach the fire.
4. Set a Reflash Watch.
5. Deenergize the space.
6. Set fire boundaries.

7. HELP
8. QUIT
: 5.

next screen display

Steve 's Score Level
6 of 8 75 %
Steve , you are the " FIRE TEAM LEADER ."

removed user session lines

FIRE TEAM LEADER, you THEN receive the following report:
All fire boundaries have been set.

FIRE TEAM LEADER, you THEN receive the following report:
A man has broken a finger while shutting the entrance
hatch door to the space with a fire.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Extinguish all the fires.
2. Obtain a satisfactory first Oxygen Test.
3. Set a Reflash Watch.
4. Choose and activate a drain method type.
5. Take care of the injured person.
6. Replace the injured person.

7. HELP
8. QUIT
: 2.

next screen display

Steve, your choice of:
Obtain a satisfactory first Oxygen Test.

has nothing to do with the current most important
action to be performed. The LAST REPORT:
A man has broken a finger while shutting the entrance
hatch door to the space with a fire.

should help you decide what to do next.

(Enter a "c." to continue.)c.

removed user session lines

next screen display

Steve 's Score Level
12 of 15 80 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER. you receive the following report:
The action: Replace the injured person.
has been reported completed.

FIRE TEAM LEADER. you THEN receive the following report:
A relief hasn't been located yet for the injured person.

FIRE TEAM LEADER. indicate your most important,
or next action to be completed.

1. Extinguish all the fires.
2. Get the oxygen testing equipment.
3. Take care of the injured person.
4. Replace the injured person.
5. Set fire boundaries.
6. Secure the Reflash Watch.

7. HELP
8. QUIT
: 4.

[next screen display]

Steve 's Score Level
15 of 18 83 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you MUST now DIRECT THE ACTIONS to complete the following TASK: Approach the fire.

FIRE TEAM LEADER. indicate your most important,
or next action to be completed.

1. Crack open the space boundary.
2. Activate both working and backup hoses
into the space.
3. Completely open the access door.
4. Store all equipment away in the Repair Locker.
5. Deenergize the space.
6. Obtain a satisfactory first Oxygen Test.

7. HELP
8. QUIT
: 5.

[next screen display]

Steve. you YOU HAVE ALREADY COMPLETED the action:
Deenergize the space.

(Enter a "c." to continue.)c.
removed user session lines

next screen display

Steve 's Score Level
19 of 23 82 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Enter the space with both the working
and backup hoses.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The space has been entered with both
the working and backup hoses.

FIRE TEAM LEADER, you receive the following report:
The action: Approach the fire.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The fire team has opened the space and is
approaching the fire.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Verify all fires are out.
2. Extinguish all the fires.
3. Set a Reflash Watch.
4. Approach the fire.
5. Perform explosive gas tests.
6. Set fire boundaries.

7. HELP
8. QUIT
: 2.

next screen display

Steve 's Score Level
20 of 24 83 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you MUST now DIRECT THE ACTIONS to complete the following TASK: Extinguish all the fires.

next screen display

* * * KNOWLEDGE TEST PROGRAM * * *

Steve, you will be shown various sequences of
operational steps to perform the Task:

Extinguish all the fires.

The reference for this task's sequence of operational steps is the
BASIC MILITARY REQUIREMENTS . NAVEDTRA 10054-E1, p.11-10

If the sequence is correct, you should enter a "yes."
otherwise a "no." if its incorrect. If your answer is incorrect,
you'll be shown the correct sequence of steps and given 0 points
for this test. If your answer is correct you'll be given 1 point
for this test.

Enter a "c." to continue.

next screen display

Steve, are the following steps the correct sequence to
perform the task:
Extinguish all the fires.

1. Activate both hoses into the space.
2. Cover all fires with water.
3. Use low velocity fog, then only if necessary
4. Use high velocity fog, then only if necessary
5. Use straight solid stream of water. REMEMBER EVERY GALLON OF WATER PUT ON A FIRE MUST BE PUMPED OVERBOARD OR DISPOSED OF IN SOME MANNER.

Enter a "yes." or "no.": yes.

Steve, your answer is correct.

Enter a "c." to continue.

removed user session lines

Steve 's Score Level
24 of 28 85 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Verify all fires are out.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
Small signs of new smoke are still being investigated
in the space.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Set a Reflash Watch.
2. Perform explosive gas tests.
3. Verify all fires are out.
4. Desmoke the space.
5. Extinguish all the fires.
6. Get the oxygen testing equipment.

7. HELP
8. QUIT
: 3.

[next screen display]
Steve 's Score Level
25 of 29 86 %
Steve , you are the " FIRE TEAM LEADER ."
[removed user session lines]

FIRE TEAM LEADER, you THEN receive the following report:
The overhaul equipment rake has broken.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Have second team member check for proper operation of the equipment.
2. Store all equipment away in the Repair Locker.
3. Perform explosive gas tests.
4. Have fire team member recheck proper operation of
malfunctioning equipment.
5. If equipment is still malfunctioning, check local area for replacement.
6. Dismiss the fire team.

7. HELP
8. QUIT
: 4.

[next screen display]
Steve 's Score Level
26 of 30 86 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Have fire team member recheck proper operation of
malfunctioning equipment.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The equipment is definitely not working correctly.
[removed user session lines]

[next screen display]
Steve 's Score Level
32 of 36 88 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Request from Repair Locker replacement equipment or an alternative
plan to correct the malfunctioning equipment situation.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
A replacement piece of equipment has been located is at the scene, and the
Repair Locker wants you to continue on your present plan of attack.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.
[removed user session lines]

[next screen display]
Steve 's Score Level
37 of 41 90 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Sample the space, which had the fire,
for explosive gases.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The explosive gas tester wasn't operated properly during the last test sample.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Sample the space, which had the fire.
   for explosive gases.
2. Dismiss the fire team.
3. Get explosive gas tester.
4. Set a Reflash Watch.
5. Obtain a satisfactory second Oxygen Test.
6. Have the explosive gas tester tested.

7. HELP
8. QUIT
: 1.

[next screen display]

Steve 's Score Level
38 of 42 90 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you MUST now DIRECT THE ACTIONS to complete the following TASK: Sample the space, which had the fire,
for explosive gases.

Enter a "c." to continue.

[next screen display]

* * * KNOWLEDGE TEST PROGRAM * * *

Steve, you will be shown various sequences of
operational steps to perform the Task:

Sample the space, which had the fire,
for explosive gases.

The reference for this task's sequence of operational steps is the
COMBUSTIBLE GAS INDICATORS, NAVEDTRA 465-08-00-82, pp. 4-5.

[removed user session lines]

[next screen display]

Steve 's Score Level
40 of 44 90 %

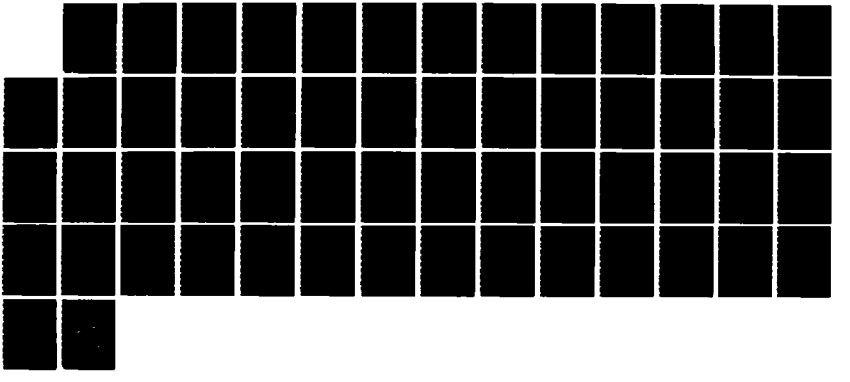
Steve , you are the " FIRE TEAM LEADER ."

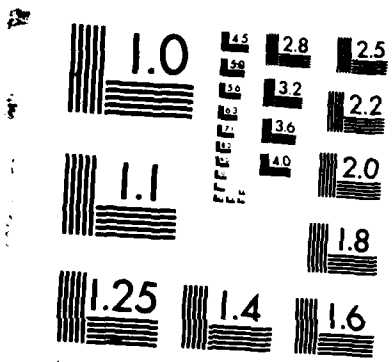
FIRE TEAM LEADER, you receive the following report:
The action: Perform explosive gas tests.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
Explosive Gas Tests indicate negative explosive gas.

AD-A173 612 DEVELOPMENT OF A SHIPBOARD DAMAGE CONTROL FIRE TEAM LEADER INTELLIGENT CO (U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA S G WEINGART 20 JUN 86 2/2

UNCLASSIFIED F/G 13/12 NL





CROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

FIRE TEAM LEADER, you THEN receive the following report:
The pre-use test of the tester indicated explosive gases
in outside the ship fresh air.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Have second team member check for proper operation of the equipment.
2. Desmoke the space.
3. If equipment is still malfunctioning, check local area for replacement.
4. Get the oxygen testing equipment.
5. Have fire team member recheck proper operation of
malfunctioning equipment.
6. Test the oxygen test equipment.
7. HELP
8. QUIT
: 4.

[next screen display]

Steve, your choice of:
Get the oxygen testing equipment.

has nothing to do with the current most important
action to be performed. The LAST REPORT:
The pre-use test of the tester indicated explosive gases
in outside the ship fresh air.

should help you decide what to do next.

[removed user session lines]

YOUR LAST REPORT WAS:
The pre-use test of the tester indicated explosive gases
in outside the ship fresh air.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Get the oxygen testing equipment.
2. Test the oxygen test equipment.
3. Have fire team member recheck proper operation of
malfunctioning equipment.
4. Desmoke the space.
5. If equipment is still malfunctioning, check local area for replacement.
6. Have second team member check for proper operation of the equipment.
7. HELP
8. QUIT
: 5.

next screen display

Steve, you have FAILED TO REALIZE that a REFLASH.
PERSONNEL INJURY or EQUIPMENT FAILURE CASUALTY has just occurred
or you've taken the WRONG IMMEDIATE ACTION!

You MUST take the CORRECT IMMEDIATE ACTIONS in the CORRECT ORDER
to CONTROL and RECOVER from this CASUALTY.

removed user session lines

next screen display

Steve 's Score Level
40 of 46 86 %
Steve , you are the " FIRE TEAM LEADER ."

YOUR LAST REPORT WAS:
The pre-use test of the tester indicated explosive gases
in outside the ship fresh air.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Desmoke the space.
2. Get the oxygen testing equipment.
3. Test the oxygen test equipment.
4. Have second team member check for proper operation of the equipment.
5. If equipment is still malfunctioning, check local area for replacement.
6. Have fire team member recheck proper operation of
   malfunctioning equipment.

7. HELP
8. QUIT
: 8.

next screen display

FINAL GRADE OF PERFORMANCE

Congratulations Steve you've survived!

Remember there's always room for improvement in Damage Control
Casualty Training. Your life, your shipmate's, and your
ship's survival may someday depend on your KNOWLEDGE as a
MAN IN CHARGE of a casualty. The following is a report of
your performance:

Casualty Task	Points	Total	Percent Grade
Overall Score	41	47	87

Deenergize the space.
Pts = 3 Total = 4 Percent Grade = 75

Set fire boundaries.
Pts = 4 Total = 4 Percent Grade = 100

Injured Person Casualty.
Pts = 3 Total = 4 Percent Grade = 75

Approach the fire.
Pts = 4 Total = 5 Percent Grade = 80

Extinguish all the fires.
Pts = 1 Total = 1 Percent Grade = 100

Perform explosive gas tests.
Pts = 5 Total = 5 Percent Grade = 100

Sample the space. which had the fire for explosive gases.
Pts = 1 Total = 1 Percent Grade = 100

Main Fire
Pts = 12 Total = 13 Percent Grade = 92

Broken Equipment Casualty.
Pts = 7 Total = 9 Percent Grade = 77

Enter a 'c.' to continue.

next screen display

FOLLOW ON TRAINING OPPORTUNITIES
For further training on the types of damage control actions or equipment operations you just directed :
Enter a 1. To Quit
2. Previous Fire Casualty Operations Training
: 1.

APPENDIX B

EXPERT LEVEL DEMONSTRATION OF COMPLEX FIRE SIMULATION

Steve, please enter the time of day(0001-2400): 1100.

Thank You. Steve, if you are not familiar with my operation, request
HELP, otherwise choose an area of Operations and/or Casualty
Control that you'd like to challenge me in!
1. HELP
2. Individual Equipment Operations and Basic Damage Control Actions
3. Individual Complex Damage Control Actions
4. Complex Fire Simulation

Enter a 1., 2., 3., or 4. for desired category: 4.

Steve, please enter your ability level as follows:
1. Beginner, 2. Advanced, 3. Expert, for this category.
Enter a 1., 2., or 3. for level:
: 3.
GOOD LUCK!

[next screen display]

DING DING DING DING DING
FIRE FIRE FIRE FIRE
FIRE IN 'E' DIVISION BERTHING , COMPARTMENT 1 - 34 - 1 - L

AWAY THE REPAIR PARY FIRE TEAM AWAY!

Steve , you are now the " FIRE TEAM LEADER ."

FIRE TEAM LEADER. indicate your most important,
or next action to be completed.

1. Extinguish all the fires.
2. Set a Reflash Watch.
3. Verify all fires are out.
4. Approach the fire.
5. Desmoke the space.
6. Test the oxygen test equipment.
7. Get to the scene equipped and ready to go.
8. Perform explosive gas tests.
9. Deenergize the space.
10. Obtain a satisfactory first Oxygen Test.
11. Set fire boundaries.
12. Get the oxygen testing equipment.

13. HELP
14. QUIT
: 3.

[next screen display]

Steve. BEFORE you can perform the actio.:
Remove all fires.

YOU MUST FIRST perform the action:
Isolate the fire.

OTHERWISE
the time and men and equipment you spend trying to remove the fires in one location will be wasted because the fire will continue to spread to new spaces until it is isolated.

(Enter a "c." to continue.)c.

[next screen display]

STUDY AND LEARN the following information to help you next time you have a similar situation.
Below is a breakdown of the specific sub-actions, which make up the action:
Isolate the fire.

1. Get to the scene equipped and ready to go.
2. Deenergize the space.
3. Set fire boundaries.

Do you want more breakdowns of these sub-actions?
INDICATE WHICH SUB-ACTIONS USING FORMAT "[#,#]."
otherwise ENTER "c." to continue.
: c.

[removed user session lines]

[next screen display]

Steve 's Score Level
0 of 1 0 %

Steve . you are the " FIRE TEAM LEADER ."

YOUR LAST REPORT WAS:
The ship has been informed of the location of the fire.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

[removed user session lines]

11. Set fire boundaries.
12. Desmoke the space.
13. HELP
14. QUIT
: 11.

[next screen display]

Steve, BEFORE you can perform the action:
Set fire boundaries.

YOU MUST FIRST perform the action:
Deenergize the space.

OTHERWISE
if the fire's source is electrical, it will continue to help the fire
grow in intensity and if the fire's source wasn't electrical,
electrical cabling in the fire space will soon breakdown due to the heat
and cause new fires.

[removed user session lines]

[next screen display]

Steve 's Score Level
0 of 2 0 %

Steve , you are the " FIRE TEAM LEADER ." .

YOUR LAST REPORT WAS:
The ship has been informed of the location of the fire.

FIRE TEAM LEADER, indicate your most important.
or next action to be completed.

[removed user session lines]

10. Get to the scene equipped and ready to go.
11. Obtain a satisfactory first Oxygen Test.
12. Deenergize the space.

13. HELP
14. QUIT
: 12.

[next screen display]

Steve, BEFORE you can perform the action:
Deenergize the space.

YOU MUST FIRST perform the action:
Get to the scene equipped and ready to go.

OTHERWISE
your personnel may deenergize the wrong spaces or risk personnel injury by not using proper safety precautions such as rubber gloves while operating breakers or fuses.

(Enter a "c." to continue.)c.

[next screen display]

STUDY AND LEARN the following information to help you next time you have a similar situation.
Below is a breakdown of the specific sub-actions, which make up the action: Get to the scene equipped and ready to go.

1. Get the fire team dressed.
2. Get the fire team to the scene.
3. Take charge of the scene.
4. Establish phone communications with the Repair Locker.
5. Charge the fire hoses.

[removed user session lines]

[next screen display]

Steve 's Score Level
0 of 3 0 %

Steve , you are the " FIRE TEAM LEADER ."

YOUR LAST REPORT WAS:
The ship has been informed of the location of the fire.

FIRE TEAM LEADER, indicate your most important, or next action to be completed.

[removed user session lines]

10. Get to the scene equipped and ready to go.
11. Verify all fires are out.
12. Get the oxygen testing equipment.

13. HELP
14. QUIT
: 10.

[next screen display]

Steve 's Score Level
1 of 4 25 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Get to the scene equipped and ready to go.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The fire team is on the scene and ready to go.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.
[removed user session lines]
7. Obtain a satisfactory first Oxygen Test.
8. Approach the fire.
9. Desmoke the space.
10. Get to the scene equipped and ready to go.
11. Extinguish all the fires.
12. Verify all fires are out.

13. HELP
14. QUIT
: 10.

[next screen display]
Steve, the LAST REPORT:
The fire team is on the scene and ready to go.

SHOULD HAVE indicated to you that the action:
Get to the scene equipped and ready to go.

has JUST BEEN COMPLETED.
[removed user session lines]

[next screen display]
Steve 's Score Level
3 of 7 42 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Have Repair Locker locate on the ship's electrical
drawings all circuit breakers which isolate
electrical power to the space on fire.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The ship's drawings haven't been located yet.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Verify all fires are out.
2. Get to the scene equipped and ready to go.
3. Dismiss the fire team.
4. Have Repair Locker locate on the ship's electrical drawings all circuit breakers which isolate electrical power to the space on fire.
5. Determine the amount of water in the space.
6. Send electrician to locate and open all circuit breakers which isolate electrical power to the space on fire.

[removed user session lines]

: 6.

[next screen display]

Steve, your PREVIOUS ACTION:

Have Repair Locker locate on the ship's electrical drawings all circuit breakers which isolate electrical power to the space on fire.

hasn't been completed yet. You SHOULD HAVE REALIZED THIS by the LAST REPORT you received:

The ship's drawings haven't been located yet.

Enter a "c." to continue.

[next screen display]

*** NEW SUB ACTION BREAKDOWN ***

Below is a breakdown of the specific sub-actions, which make up the action:
Have Repair Locker locate on the ship's electrical drawings all circuit breakers which isolate electrical power to the space on fire.

[removed user session lines]

[next screen display]

Steve 's Score Level
5 of 11 45 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Send electrician to locate and open all circuit
breakers which isolate electrical power to the
space on fire.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The electrician has open all circuit breakers
which provide power to the space.

FIRE TEAM LEADER, you receive the following report:
The action: Deenergize the space.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The space has not yet been deenergized.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.
removed user session lines
10.Deenergize the space.
11.Set a Reflash Watch.
12.Extinguish all the fires.

13. HELP
14. QUIT
: 10.
removed user session lines

next screen display
FINAL GRADE OF PERFORMANCE
removed user session lines

Casualty Task	Points	Total	Percent Grade
Overall Score	7	13	53

Have Repair Locker locate on the ship's electrical, drawings all circuit breakers which isolate, electrical power to the space on fire.
Pts = 0 Total = 1 Percent Grade = 0

Deenergize the space.
Pts = 3 Total = 4 Percent Grade = 75

Main Fire
Pts = 4 Total = 8 Percent Grade = 50

Enter a 'c.' to continue.

next screen display

FOLLOW ON TRAINING OPPORTUNITIES
For further training on the types of damage control actions or equipment operations you just directed :
Enter a 1. To Quit
2. Previous Fire Casualty Operations Training
: 2.

next screen display

* * * PREVIOUS TRAINING REVIEW * * *

Steve, you will be retested on the tasks which you were just tested on. I've taken the liberty of ordering the tasks with your best first, then your worst second, then your second best third, then your second worst fourth and so on.
I think you'll enjoy this order of retraining the best.
GOOD LUCK!

Enter a "c." to continue.

next screen display

Steve, your task will be:
Deenergize the space.
Your previous performance grade was 75 for this task.
removed user session lines

next screen display

Steve 's Score Level
2 of 2 100 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Send electrician to locate and open all circuit breakers which isolate electrical power to the space on fire.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The electrician has open all circuit breakers which provide power to the space.

THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:

Deenergize the space.

Enter a "c." to continue.

next screen display

Steve, your task will be:
Have Repair Locker locate on the ship's electrical. drawings all circuit breakers which isolate. electrical power to the space on fire.

Your previous performance grade was 0 for this task.
removed user session lines

next screen display

THANK YOU FOR USING THE

"DAMAGE CONTROL"

"FIRE TEAM LEADER"

LEARNING CENTER

For more training START THE PROGRAM OVER.

APPENDIX C

DEMONSTRATION OF FIRE'S SECONDARY MODES OF OPERATION

WELCOME TO THE

"DAMAGE CONTROL"

"FIRE TEAM LEADER"

LEARNING CENTER

PROLOG requires all input to be followed by a period(".").

BEGINNER TEST RUN SCREEN DISPLAY

Welcome. I am an EXPERT in FIRE FIGHTING and DAMAGE CONTROL!
I am ready to challenge your knowledge as a fire team scene
leader or supervisor! Please enter your NAME: 'Steve'.

Steve, please enter the time of day(0001-2400): 0930.

Thank You. Steve, if you are not familiar with my operation, request
HELP. otherwise choose an area of Operations and/or Casualty
Control that you'd like to challenge me in!
1. HELP
2. Individual Equipment Operations and Basic Damage Control Actions
3. Individual Complex Damage Control Actions
4. Complex Fire Simulation

Enter a 1., 2., 3., or 4. for desired category: 2.

Steve, please enter your ability level as follows:
1. Beginner. 2. Advanced. 3. Expert, for this category.
Enter a 1., 2., or 3. for level:
: 1.
GOOD LUCK!

next screen display

* * * Equipment Operation Tests * * *

Steve, you will be shown a list of equipment related operations
and asked to enter which operations you want to review and be tested
on. A running score of your performance on your chosen equipment
operations will be displayed. -- GOOD LUCK!

next screen display

1. Establish phone communications with the Repair Locker.
2. Charge the fire hoses.
3. Have Repair Locker locate on the ship's electrical drawings all circuit breakers which isolate electrical power to the space on fire.
4. Send fire boundary teams to each adjacent space with portable fire equipment in order to verify the boundaries are containing the fire.
5. Have personnel check their OBA hoses for proper connections.
6. Have personnel strap their OBA's firmly onto their bodies.
7. Have personnel install their OBA cannisters.
8. Have personnel don their OBA face pieces.
9. Identify the deck, frame, centerline relationship, and function of the space.
10. Determine the route to the scene.
11. Activate both working and backup hoses into the space.
12. Extinguish all the fires.
13. Set a Reflash Watch.
14. Have the explosive gas tester tested.
15. Sample the space, which had the fire, for explosive gases.
16. Choose a desmoke method.
17. Have a smoke removal path established.
18. Test the oxygen test equipment.
19. Obtain a satisfactory first Oxygen Test.
20. Obtain a satisfactory second Oxygen Test.
21. Determine the amount of water in the space.
22. Choose and activate a drain method type.
23. Conduct a debriefing of the fire casualty with the fire team.
24. Remove the injured person to a safe location.
25. Have basic first aid administered to the injured person.

Steve, INDICATE WHICH OPERATIONS USING FORMAT "[#,#]."
ie. [1,3,4].
: [18,12,25].

next screen display

* * * KNOWLEDGE TEST PROGRAM * * *

Steve, you will be shown various sequences of
operational steps to perform the Task:

Test the oxygen test equipment.

The reference for this task's sequence of operational steps is the
FLAME SAFETY LAMP, NAVEDTRA 465-08-00-82, pp. 9-10.

If the sequence is correct, you should enter a "yes."
otherwise a "no." if its incorrect. If your answer is incorrect,
you'll be shown the correct sequence of steps and given 0 points
for this test. If your answer is correct you'll be given 1 point
for this test.

Enter a "c." to continue.

[next screen display]

Steve, are the following steps the correct sequence to
perform the task:
Test the oxygen test equipment.

[removed user session lines]

[next screen display]

Steve, are the following steps the correct sequence to
perform the task:
Test the oxygen test equipment.

1. Blow against the glass and gaskets to test the lamp for leakage.
   Leakage wil cause the flame to flicker.
2. Pull the knob down and adjust the flame to approximately
   3/8 inch in height. Allow the flame to burn about 5 minutes to reach
   its normal operating temperature.
3. Push on the relighter handle(igniter) and slide it up as
   far as it will go. Turn the relighter handle and ignite the wick.
4. Turn up the wick using the wick adjuster.

Enter a "yes." or "no.": yes.

Steve, you are incorrect, the above sequence
is not correct! Below is the correct sequence,
STUDY AND LEARN it before you continue.

1. Turn up the wick using the wick adjuster.
2. Push on the relighter handle(igniter) and slide it up as far as it will go. Turn the relighter handle and ignite the wick.
3. Pull the knob down and adjust the flame to approximately 3/8 inch in height. Allow the flame to burn about 5 minutes to reach its normal operating temperature.
4. Blow against the glass and gaskets to test the lamp for leakage. Leakage will cause the flame to flicker.

Enter a "c." to continue.

[next screen display]

Steve 's Score Level
0 of 1 0 %

* * * KNOWLEDGE TEST PROGRAM * * *

Steve, you will be shown various sequences of operational steps to perform the Task:

Extinguish all the fires.

The reference for this task's sequence of operational steps is the
BASIC MILITARY REQUIREMENTS , NAVEDTRA 10054-E1. p.11-10

[removed user session lines]

Steve. are the following steps the correct sequence to perform the task:
Extinguish all the fires.

1. Activate both hoses into the space.
2. Cover all fires with water.
3. Use low velocity fog, then only if necessary
4. Use high velocity fog. then only if necessary
5. Use straight solid stream of water. REMEMBER EVERY GALLON OF WATER PUT ON A FIRE MUST BE PUMPED OVERBOARD OR DISPOSED OF IN SOME MANNER.

Enter a "yes." or "no.": no.

Steve. you are incorrect. the above sequence is correct!
STUDY AND LEARN it before you continue.

Enter a "c." to continue.

next screen display

* * * KNOWLEDGE REVIEW * * *

Steve, you will be shown a review of
important knowledge required in the performance of Task:

Have basic first aid administered to the injured person.

STUDY AND LEARN THIS INFORMATION and for further guidance refer to reference:
BASIC MILITARY REQUIREMENTS, NAVEDTRA 10054-E1, pp. 18-1.2.
you will not be graded on this review.

Enter a "c." to continue.

[next screen display]

GENERAL FIRST AID RULES

Although each case involving injury or sickness presents its own special problems, there are some general rules, given here, that apply to practically all situations. Become familiar with these basic rules before you go on to learn first aid treatment for specific types of injuries.

1. Keep the victim lying down, head level with the body, until you have found out what kind of injury has occurred and how serious it is. If the victim shows one of the following difficulties, however, follow the rule given for that specific problem:

[removed user session lines]

8. Keep the injured person comfortably warm-warm enough to maintain normal body temperature.

[removed user session lines]

[next screen display]

Steve, please enter the time of day(0001-2400): 0945.

Thank You, Steve, if you are not familiar with my operation, request HELP, otherwise choose an area of Operations and/or Casualty Control that you'd like to challenge me in!

1. HELP
2. Individual Equipment Operations and Basic Damage Control Actions
3. Individual Complex Damage Control Actions
4. Complex Fire Simulation

Enter a 1., 2., 3., or 4. for desired category: 3.

Steve, please enter your ability level as follows:
1. Beginner, 2. Advanced, 3. Expert, for this category.
Enter a 1., 2., or 3. for level:
: 1.
GOOD LUCK!

next screen display

* * * DAMAGE CONTROL ACTION TESTS * * *

Steve, you will be shown a list of damage control related actions and asked to enter which actions you want to review and be tested on. A running score of your performance on your chosen damage control actions will be displayed. -- GOOD LUCK!

Enter a " " to continue.

next screen display

1. Injured Person Casualty.
2. Broken Equipment Casualty.
3. Get to the scene equipped and ready to go.
4. Deenergize the space.
5. Set fire boundaries.
6. Get the fire team dressed.
7. Get the fire team to the scene.
8. Take charge of the scene.
9. Send fire boundary teams to shut all physical openings to the space with the fire.
10. Dress the fire team in Battle Dress.
11. Dress the fire team in OBA's.
12. Equip the fire team with fire fighting equipment.
13. Have personnel put their helmets on.
14. Determine the required fire fighting equipment for personnel.
15. Locate the fire.
16. Relieve the senior person in charge at the scene.
17. Approach the fire.
18. Crack open the space boundary.
19. Verify all fires are out.
20. Perform explosive gas tests.
21. Desmoke the space.
22. Store all equipment away in the Repair Locker.
23. Secure the Reflash Watch.
24. Take care of the injured person.
25. Have medical aid given to the injured person.

Steve. INDICATE WHICH ACTIONS USING FORMAT "[#.#]."
ie. [1.3.4].
: [23.3.8.20].

next screen display

* * * DAMAGE CONTROL ACTION TEST * * *

Steve, you may now direct the actions to
properly execute the action:

Secure the Reflash Watch.

Your score for this task will be displayed as you carry
out the actions. -- GOOD LUCK!

Enter a "c." to continue.

[next screen display]

FIRE TEAM LEADER. indicate your most important.
or next action to be completed.

1. Check that the required Reflash Watch
   on station time has elapsed.
2. Have the Reflash Watch put his equipment
   away in the Repair Locker.
3. Get the Enginneer's or his designated representative's
   permission to secure the Reflash Watch.

4. HELP
5. QUIT.

: 1.

[next screen display]

Steve 's Score Level
1 of 1 100 %

Steve , you are the " FIRE TEAM LEADER ."

removed user session lines

[next screen display]

Steve 's Score Level
2 of 2 100 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER. you receive the following report:
The action: Get the Enginneer's or his designated representative's
permission to secure the Reflash Watch.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The Engineer has given his permission
to secure the reflash watch.

FIRE TEAM LEADER, indicate your most important.
or next action to be completed.

1. Have the Reflash Watch put his equipment
   away in the Repair Locker.
2. Dismiss the Reflash Watch watchstander.
3. Get the Enginneer's or his designated representative's
   permission to secure the Reflash Watch.
4. HELP
5. QUIT
: 3.

[next screen display]

Steve, the LAST REPORT:
The Engineer has given his permission
to secure the reflash watch.

SHOULD HAVE indicated to you that the action:
Get the Enginneer's or his designated representative's
permission to secure the Reflash Watch.

has JUST BEEN COMPLETED.
The following action:
Get the Enginneer's or his designated representative's
permission to secure the Reflash Watch.

is a basic level action which doesn't simplify
into subactions for further LEARNING.

[removed user session lines]

[next screen display]

Steve 's Score Level
5 of 6 83 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Dismiss the Reflash Watch watchstander.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The Reflash Watch has been secured.

THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:
Secure the Reflash Watch.

next screen display

* * * DAMAGE CONTROL ACTION TEST * * *

Steve, you may now direct the actions to
properly execute the action:

Get to the scene equipped and ready to go.

Your score for this task will be displayed as you carry
out the actions. -- GOOD LUCK!

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Take charge of the scene.
2. Get the fire team to the scene.
3. Get the fire team dressed.

4. HELP
5. QUIT
: 3.

next screen display

Steve 's Score Level
1 of 1 100 %
Steve , you are the " FIRE TEAM LEADER ."
removed user session lines

FIRE TEAM LEADER, you receive the following report:
The action: Get the fire team to the scene.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The fire team is at the scene.

FIRE TEAM LEADER, indicate your most important,
or next action to be completed.

1. Get the fire team to the scene.
2. Take charge of the scene.
3. Establish phone communications with
the Repair Locker.

4. HELP
5. QUIT
: 3.

next screen display

Steve, BEFORE you can perform the action:
Establish phone communications with
the Repair Locker.

YOU MUST FIRST perform the action:
Take charge of the scene.

OTHERWISE
you will only cause chaos and confusion at the scene for the personnel who are presently attempting to keep the fire under control. You MUST relieve the on scene leader, prior to having your fire team start any actions at the scene.

(Enter a "c." to continue.)c.

next screen display

STUDY AND LEARN the following information to help you next time you have a similar situation.
Below is a breakdown of the specific sub-actions, which make up the action:
Take charge of the scene.

1. Visually assess the status of the casualty control procedures in progress.
2. Identify the senior person in charge at the scene.
3. Relieve the senior person in charge at the scene.

removed user session lines

next screen display

7 of 9 77 %

Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Charge the fire hoses.
has been reported completed.

THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:
Get to the scene equipped and ready to go.

Enter a "c." to continue.

next screen display

* * * DAMAGE CONTROL ACTION TEST * * *

Steve, you may now direct the actions to
properly execute the action:
Take charge of the scene.

removed user session lines

next screen display

Steve 's Score Level
5 of 6 83 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Relieve the senior person in charge at the scene.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
You are now the man in charge at the scene.

THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:

Take charge of the scene.

Enter a "c." to continue.

next screen display

* * * DAMAGE CONTROL ACTION TEST * * *

Steve, you may now direct the actions to
properly execute the action:

Perform explosive gas tests.

Your score for this task will be displayed as you carry
out the actions. -- GOOD LUCK!

removed user session lines

Steve 's Score Level
5 of 5 100 %
Steve , you are the " FIRE TEAM LEADER ."

FIRE TEAM LEADER, you receive the following report:
The action: Sample the space, which had the fire,
for explosive gases.
has been reported completed.

FIRE TEAM LEADER, you THEN receive the following report:
The explosive gas tests indicate negative explosive gases.

THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:

Perform explosive gas tests.

Enter a "c." to continue.

APPENDIX D

FIRE'S MAJOR SUPPORTING PROGRAM FILES

File Name: fire

FIRE: A FIRE TEAM LEADER LEARNING CENTER SYSTEM

MODE OF OPERATION: COMPLEX FIRE CASUALTY

AUTHOR: LT. Steve Weingart, USN

PROGRAM INITIATOR

```
fire:- consult(introduction),intro(Cat),abolish(intro,1),
    standby,consult(node_actions),standby,
    consult(action_results),standby,consult(execute_tp_order),standby,
    consult(execute_tp_info),standby,
    consult(badconsequences),standby, initiate(Cat).

initiate(4):- sourcefile(_,4,Filename,Type),
    consult(Filename),clear, execute(Filename,Type).interface(grade).
initiate(Category):- sourcefile(_,Category,Filename,Type),
    consult(Filename),clear, execute(Filename,Type).

standby:- clear,blank_lines(4),
    writelist(29,['"DAMAGE CONTROL"']),blank_lines(3),
    writelist(29,['"FIRETEAM LEADER"']),blank_lines(3),
    writelist(29,[' LEARNING CENTER']), blank_lines(4),
    writelist(21,['PLEASE STANDBY WHILE LOADING FILES']), blank_lines(2),
    writelist(9,['REMEMBER to "PAUSE THE SCREEN MOVEMENT" ',
    ' with the "NO SCROLL" key,']),
    writelist(9,['AND PROLOG requires all input to be followed',
    ' by a period("."). ']).

welcome:- clear, blank_lines(4),
    writelist(30,['WELCOME TO THE']), blank_lines(3),
    writelist(29,['"DAMAGE CONTROL"']),blank_lines(3),
    writelist(29,['"FIRETEAM LEADER"']),blank_lines(3),
    writelist(29,[' LEARNING CENTER']), blank_lines(4),
    writelist(9,['REMEMBER to "PAUSE THE SCREEN MOVEMENT" ',
    'with the "NO SCROLL" key,']),
    writelist(9,['AND PROLOG requires all input to be followed ',
    'by a period("."). ']), standby7, clear.
```

```
goodby:- clear, blank_lines(5).
    writelist(27, 'THANK YOU FOR USING THE'), blank_lines(3).
    writelist(29, '"DAMAGE CONTROL"' ).blank_lines(3).
    writelist(29, '"FIRETEAM LEADER"' ).blank_lines(3).
    writelist(29,' LEARNING CENTER'), blank_lines(2),
    writelist(17,' For more training START THE PROGRAM OVER.'),
    standby7, clear, halt.

standby7:- system("sleep 7").
standby7.
```

PRIMARY SOURCE FILES

```
sourcefile(mainfire,1,help,tp).
sourcefile(mainfire.2,equipmentops,tp).
sourcefile(mainfire.3,dcactions,tp).
sourcefile(mainfire.4,mainfire,ex).
sourcefile(mainfire,5,prevtraining,tp).

sourcefile(mainfire,[1],reflash,ex).
sourcefile(mainfire,[2],injuredperson,ex).
sourcefile(mainfire,[3],brokenequip.ex).
```

PRIMARY AUXILARY PREDICATES

```
append(X,[],[X]):- !.
append(X,[Y|List],[Y|NewList]):- append(X,List,NewList).

blank_lines(0):- !.
blank_lines(N):- nl, N1 is N - 1, blank_lines(N1).

blank_spaces(0):- !.
blank_spaces(N):- write(' '), N1 is N-1, blank_spaces(N1),!.

clear:- system("clear").
clear.

continue:- nl, nl, write('Enter a "c." to continue.'), read(C), clear, !.

deleteone(X,[],[]).
deleteone(X,[X|List],List):- !.
deleteone(X,[Y|List],[Y|NewList]):- deleteone(X,List,NewList).
```

```
delete_same(_,_,_,_):- !.
delete_same(X,_,X,_):- !.
delete_same(_,X,_,X):- !.
delete_same(X,Y,NewX1,NewY1):- member(Z,Y), member(Z,X), deleteone(Z,Y,NewY),
    deleteone(Z,X,NewX), delete_same(NewX,NewY,NewX1,NewY1), !.
delete_same(W,X,W,X):- !.

describe_task([T],[T]):- !.
describe_task(Task,TaskDesc):- sourcefile(_,NodeNo,Task,_),
    node_action(NodeNo,TaskDesc,_), !.
describe_task(Task,TaskDesc):- assign([Task],TaskDesc),!.

first([X|List],X):- !.

mapfirst([[L1,L2]|RestL],[L1|List]):- mapfirst(RestL,List).

mapsecond([[L1,L2]],[L2]):- !.
mapsecond([[L1,L2]|RestL],[L2|List]):- mapsecond(RestL,List), !.

max_task([X],X).
max_task([[Task,Pts,Total,Percent]|List],[Task,Pts,Total,Percent]):-
    max_task(List,[_,_,_,Max]), Percent >= Max.
max_task([[Task,Pts,Total,Percent]|List],[Task1,Pts1,Total1,Max]):-
    max_task(List,[Task1,Pts1,Total1,Max]).

member(X,[]):- !,fail.
member(X,[X|_]).
member(X,[_|Y]):- member(X,Y).

min_task([X],X).
min_task([Task,Pts,Total,Percent|List],[Task,Pts,Total,Percent]):-
    min_task(List,[_,_,_,Min]), Percent =< Min.
min_task([Task,Pts,Total,Percent|List],[Task1,Pts1,Total1,Min]):-
    min_task(List,[Task1,Pts1,Total1,Min]).

percent(Pts,Total,Percent):- Percent is (Pts/Total)*100.



pr_list_lines(_,[]):- !.
pr_list_lines(Index,[H|T]):- blank_spaces(Index),write(H), nl,
    pr_list_lines(Index,T).
```

```
pr_list_on_line([]):- nl,!.
pr_list_on_line([H|T]):- write(H), tab(1), pr_list_on_line(T),!.

score_update(Task,Score):- describe_task(Task,TaskDesc).
    retract(overall_score(Pts1,Total1,_)),
    Pts is Pts1 + Score, Total is Total1 + 1, percent(Pts,Total,Per).
    asserta(overall_score(Pts,Total,Per)), interface(score,Pts,Total,Per),
    task_score_update(TaskDesc,Score).
score_update(Task,1):- describe_task(Task,TaskDesc),
    asserta(overall_score(1,1,100)),
    interface(score,1,1,100), task_score_update(TaskDesc,1).
score_update(Task,0):- describe_task(Task,TaskDesc),
    asserta(overall_score(0,1,0)),
    interface(score,0,1,0), task_score_update(TaskDesc,0).


select([X|_],1,X).
select([X|Y],N,Member):- N1 is N - 1, select(Y,N1,Member).

task_score_update(Task,Score):- retract(task_score(Task,Pts1,Total1,_)).
    Pts is Pts1 + Score, Total is Total1 + 1, percent(Pts,Total,Percent),
    assertz(task_score(Task,Pts,Total,Percent)).
task_score_update(Task,1):- assertz(task_score(Task,1,1,100)).
task_score_update(Task,0):- assertz(task_score(Task,0,1,0)).

write_no_list(List,Index,Len):-  member(Item,List),
    write(Index), write('. '), writelist(0,Item),
    deleteone(Item,List,NewList), Index1 is Index + 1.
    not(Index1>Len), write_no_list1(NewList,Index1,Len), !.

write_no_list1(List,Index,Len):- write_no_list2(List,Index,Len).
write_no_list1(List,Index,Len).

write_no_list2(List,Index,Len):- member(Item,List),
    write(Index), write('. '), writelist(0,Item),
    deleteone(Item,List,NewList), Index1 is Index + 1, !,
    not(Index1>Len), write_no_list2(NewList,Index1,Len).

writelist(Index,List):- pr_list_lines(Index,List),!.
```

GAME ENVIRONMENT

RANDOM TASK GENERATOR

```
random(2,N):- retract(seed(S)). N1 is (S mod 3) - 1. N is (N1 mod 2) + 1,
    Newseed is (125 * S - 1) mod 4096, asserta(seed(Newseed)).!.

random(R,N):- retract(seed(S)), N is (S mod R) + 1,
    Newseed is (125 * S - 1) mod 4096, asserta(seed(Newseed)).!.

random_yes_no(CasFac,YesNo):- random(100,N). experience_level(ExpLev,_),
    Level is ExpLev - CasFac,((N < Level,performtask(no,YesNo));
    performtask(yes,YesNo)).!.

performtask(X,X).
```

TASK EXECUTOR

```
execute(Task,ex):- start(Task,Start),check_casualty_code,
    correct_action_order(Start,Task), role(Task,Role),
    check_last_report(Task),
    abolish(active_role,1), asserta(active_role(Role)).
    successor(Task,Start,Newstate),
    asserta(previous_node(Task,Start)),action_seq(Task,Newstate,[Start]), !.
execute(Task,tp):- tp_file_type(_,Task,Type), execute_tp(Task,Type), !.
execute(Task,nex).

check_casualty_code:- handlecasualty_code(HC).
check_casualty_code:- asserta(handlecasualty_code(0)).

check_last_report(Task):- last_report(Task,Report).
check_last_report(Task):- retract(last_report(_,Report)),
    asserta(last_report(Task,Report)).
check_last_report(_).

action_seq(Task,complete,Statelist).
action_seq(Task,State,Statelist):- not(member(State,Statelist)).
    test_user(Task,State,Statelist). action(Task,State,Statelist).
action_seq(Task,State,Statelist):- test_user(Task,State,Statelist).
    update_results(Task,State,Statelist,Nstatelist),remove_reflash,
    action(Task,State,Nstatelist).
```

```
action(Task,State,Statelist):- node_action(State,_,Type),
    complete_action(Task,State,Type),
    action_results(Task,State,HC,Cas),
    (var(Cas); complete_casualty(Task,State,Nstate,HC,Cas)),
    ((var(Nstate),successor(Task,State,Nstate));true),
    action_seq(Task,Nstate,[State|Statelist]).


complete_action(Task,State,nex):- interface(comp_action,Task,State).
complete_action(Task,State,_):- casfac(Task,State,Casfac),
    random_yes_no(Casfac,YesNo), YesNo = no,
    interface(comp_action,Task,State).
complete_action(Task,State,Type):- interface(ncomp_action,Task,State),
    sourcefile(Task,State,File,Type), consult(File),
    ((Type = tp,continue);true),
    nl, execute(File,Type),
    interface(comp_action,Task,State).

complete_casualty(Task,_,_,HC,Cas):- casfac(_,HC,Casfac),
    random_yes_no(Casfac,YesNo), YesNo = no.
complete_casualty(Task,State,Nstate,HC,Cas):- random(2,N),
    select(Cas,N,Casualty),
    asserta(result(Task,State,HC,Casualty)),
    interface(announce_cas,HC,Casualty),
    check_casualty_code,
    retract(handlecasualty_code(OldHC)), asserta(handlecasualty_code(HC)),
    sourcefile(_,[HC],File,Type),
    ((reflash(HC), assign([1,1,2,2],Nstate));
    ( consult(File), nl,
    execute(File,Type), cleandatabase(_,[HC],Type),
    retract(handlecasualty_code(OoldHC)),retract(last_report(File,Rep)),
    asserta(handlecasualty_code(0)))).

reflash(1).

assign(X,X).

casfac(Task,CasNo,Casfac):-
     task_score(CasNo,Pts,Total,Percent),
    ((Percent < 90, Casfac is Total - Pts, (Casfac =< 25; Casfac is 25));
    (Casfac is -15)),!.
casfac(_,_,0):- !.

update_results(Task,X,[X|List],List).
update_results(Task,X,[Y|List],[Statelist]):- retract(result(Task,Y,_,_)).
    update_results(Task,X,List,Statelist).
```

```
remove_reflash:- handlecasualty_code(1),retract(handlecasualty_code(1)).
    retract(last_report(HC,Rep)),asserta(handlecasualty_code(0)).
remove_reflash.

correct_action_order(Start,Task):- bagof([X,Y],successor(Task,X,Y),BagList).
    mapfirst(BagList,List). assertz(successor(Task,State,State)),
    makeunique(List,Ulist). deleteone(Start,Ulist,UniqueList),
    asserta(action_order(Task,UniqueList)).

test_user(Task,State,Statelist):- action_order(Task,ActionList),
    exp_action_list(State,ActionList,ExpList),
    test(Task,State,ExpList), check_save_dummy_items.

exp_action_list(State,List,ExpList):- experience_level(Exp,_),
    difficulty(Exp,Dif), split_list(Dif,State,List,ActionList),
    length(ActionList, Len),
    ((Len < Dif, extend(Len,Dif,State,ActionList,ExpList));
    (true,assign(ActionList,ExpList))).

difficulty(90,3).
difficulty(75,6).
difficulty(60,12).

split_list(3,X,[X,A],[X,A]).
split_list(3,X,[A,X],[A,X]).
split_list(3,X,[X,A,B|L],[X,A,B]).
split_list(3,X,[A,X,B|L],[A,X,B]).
split_list(3,X,[A,B,X],[A,B,X]).
split_list(3,X,[Y|List],ActionList):- split_list(3,X,List,ActionList).

split_list(6,X,[X,A,B,C,D,E|L],[X,A,B,C,D,E]).
split_list(6,X,[A,X,B,C,D,E|L],[A,X,B,C,D,E]).
split_list(6,X,[A,B,X,C,D,E|L],[A,B,X,C,D,E]).
split_list(6,X,[A,B,C,X,D,E],[A,B,C,X,D,E]).
split_list(6,X,[A,B,C,D,X,E],[A,B,C,D,X,E]).
split_list(6,X,[A,B,C,D,E,X],[A,B,C,D,E,X]).
split_list(6,X,[Y|List],ActionList):- length(List,Len), Len > 6,
    split_list(6,X,List,ActionList).
split_list(6,X,List,ActionList):- split_list(3,X,List,ActionList).
```

```
split_list(12,X,X,A,B,C,D,E,F,G,H,I,J,K,List,X,A,B,C,D,E,F,G,H,I,J,K).
split_list(12,X,A,X,B,C,D,E,F,G,H,I,J,K,List,A,X,B,C,D,E,F,G,H,I,J,K).
split_list(12,X,A,B,X,C,D,E,F,G,H,I,J,K,List,A,B,X,C,D,E,F,G,H,I,J,K).
split_list(12,X,A,B,C,X,D,E,F,G,H,I,J,K,List,A,B,C,X,D,E,F,G,H,I,J,K).
split_list(12,X,A,B,C,D,X,E,F,G,H,I,J,K,List,A,B,C,D,X,E,F,G,H,I,J,K).
split_list(12,X,[A,B,C,D,E,X,F,G,H,I,J,K],[A,B,C,D,E,X,F,G,H,I,J,K]).
split_list(12,X,[A,B,C,D,E,F,X,G,H,I,J,K],[A,B,C,D,E,F,X,G,H,I,J,K]).
split_list(12,X,[A,B,C,D,E,F,G,X,H,I,J,K],[A,B,C,D,E,F,G,X,H,I,J,K]).
split_list(12,X,[A,B,C,D,E,F,G,H,X,I,J,K],[A,B,C,D,E,F,G,H,X,I,J,K]).
split_list(12,X,[A,B,C,D,E,F,G,H,I,X,J,K],[A,B,C,D,E,F,G,H,I,X,J,K]).
split_list(12,X,[A,B,C,D,E,F,G,H,I,J,X,K],[A,B,C,D,E,F,G,H,I,J,X,K]).
split_list(12,X,[A,B,C,D,E,F,G,H,I,J,K,X],[A,B,C,D,E,F,G,H,I,J,K,X]).
split_list(12,X,[Y|List],ActionList):- length([Y|List],Len), Len >12,
    split_list(12,X,List,ActionList).
split_list(12,X,List,ActionList):- split_list(6,X,List,ActionList).


extend(X,X,State,List,List).
extend(Len,Dif,State,List,NewList):-  action_order(mainfire,Actions),
    random(20,N).
    select(Actions,N,Dummy),not(compare_(Dummy,State,above)),
    savedummymenuitems(Dummy),
    append(Dummy,List,Nlist), Len1 is Len - 1,
    extend(Len1,Dif,State,Nlist,NewList).

savedummymenuitems(Dummy):- not(saveextendeditems(ExtItems)),
    asserta(saveextendeditems([Dummy])).
savedummymenuitems(Dummy):- retract(saveextendeditems(ExtItems)),
    append(Dummy,ExtItems,NExtItems),asserta(saveextendeditems(NExtItems)).


check_save_dummy_items:- retract(saveextendeditems(DummyItems)).
check_save_dummy_items.

makeunique([],[]).
makeunique([X|List],UniqueList):- var(X), makeunique(List,UniqueList).
makeunique([X|List],UniqueList):- member(X,List), makeunique(List,UniqueList).
makeunique([X|List],[X|UniqueList]):- makeunique(List,UniqueList).
```

STUDENT EXPERT CHOICE COMPARER

```
test(Task,State,ExpList):-
    interface(menu,ExpList,UserAction).
    ((UserAction = State, score_update(Task,1),retract(previous_node(Task,Prev)),
      asserta(previous_node(Task,State))):
     (previous_node(Task,PrevNode),last_report(Task,Report),
     handlecasualty_code(HC),
     retract(seed(S)), S1 is S +1, asserta(seed(S1)),
    determine_SM(PrevNode,UserAction,HC,ForgotAction,Case),
    teaching_module(UserAction,ForgotAction,Report,Case), score_update(Task,0)),
    nl,write('YOUR LAST REPORT WAS: '), nl, writelist(0,Report),nl,nl.
    test(Task,State,ExpList)).
```

STUDENT MODEL

```
determine_SM(_,User,HC,_,0):- saveextendeditems(DummyItems),
    member(User,DummyItems), not(HC = 0).
determine_SM(_,_,HC,_,1):- not(HC= 0).
determine_SM(Prev,User,_,_,2):- compare_(User,Prev,less).
determine_SM(Prev,User,_,_,3):- compare_(Prev,User,equal).
determine_SM(PrevAction,UserAction,HC,MostGeneralAction,4):-
    wanted(PrevAction,UserAction,ForgotNodes),
    maplength(ForgotNodes,NewForgotNodes),
    minlength(NewForgotNodes,Len), not(Len = 0),
    collect_shortest(Len,NewForgotNodes,MostGeneral),
    greatestmostgeneralnode(MostGeneral,MostGeneralAction).
determine_SM(_,_,_,_,5).

compare_([],[],equal):- !.
compare_([],L,above):- !.
compare_([X],[],directlybelow):- !.
compare_(L,[],below):- !.
compare_([X|L1],[Y|L2],greater):- X>Y,!.
compare_([X|L1],[Y|L2],less):- X<Y, !.
compare_([X|L1],[X|L2],A):- compare_(L1,L2,A). !.

wanted(P,U,N):- bagof(X,wanteditem(P,U,X),N), !.
wanteditem(P,U,M):- node_action(M,_,_),compare_(M,P,greater), compare_(M,U,less).

maplength([],_):- !, fail.
maplength([X],[[X,Y]]):- length(X,Y),!.
maplength([X|L1],[[X,Y]|L2]):- length(X,Y), maplength(L1,L2), !.
```

```
minlength( X,Y ,Y):- !.
minlength( X,Y L1 ,Z):- minlength(L1,Z). Z=<Y,!.
minlength( X,Y L1 ,Y):- minlength(L1,Z). Y<Z. !.

collect_shortest(Len,List, _):- asserta(shortitems([])).member( X,Len ,List).
    retract(shortitems(SI)). append( X,Len ,SI, ShortItems).
    asserta(shortitems(ShortItems)).fail.
collect_shortest(Len,List,ShortItems):- retract(shortitems(ShortItems)).

greatestmostgeneralnode([X, _],X).
greatestmostgeneralnode(ListGenNodes, _):- asserta(greatest([0])),
    member([X, _],ListGenNodes),member([Y, _],ListGenNodes), compare_(X,Y,greater).
    retract(greatest(G)). ((compare_(X,G,greater). asserta(greatest(X)));
    asserta(greatest(G))), fail.
greatestmostgeneralnode( _,MostGeneralNode):- retract(greatest(MostGeneralNode)).
```

TUTORING MODULE

```
teaching_module(State, _,Report,0):- user(Name).clear,
    write(Name), write(', your choice of: '),nl,
    node_action(State,StateDesc, _),writelist(0,StateDesc),nl,nl.
    write('has nothing to do with the current most important').nl,
    write('action to be performed. The LAST REPORT: ').nl.
    writelist(0,Report),nl,nl,
    write('should help you decide what to do next.'). blank_lines(4),
    write('(Enter a "c." to continue.)'), read(C),clear, try_again_message.

teaching_module( _, _, _,1):- user(Name), clear.
    write(Name), write(', you have FAILED TO REALIZE that a REFLASH. '). nl.
    write('PERSONNEL INJURY or EQUIPMENT FAILURE CASUALTY
    has just occurred').
    nl,write('or you''ve taken the WRONG IMMEDIATE ACTION!').nl,nl.
    write('You MUST take the CORRECT IMMEDIATE ACTIONS in the
    CORRECT ORDER'),
    nl. write('to CONTROL and RECOVER from this CASUALTY.'), nl,
    blank_lines(2). display_node_breakdown(State,0). try_again_message.

teaching_module(State, _, _,2):- user(Name), clear,
    write(Name), write(', you YOU HAVE ALREADY COMPLETED the action: ').
    nl. node_action(State,StateDesc, _). writelist(0,StateDesc),
    blank_lines(4).
    write('(Enter a "c." to continue.)'). read(C).clear, try_again_message.
```

```
teaching_module(State,_,Report,3):- user(Name), clear,
    write(Name), write(', the LAST REPORT: '), nl,
    writelist(0,Report), nl,
    write('SHOULD HAVE indicated to you that the action:'), nl,
    node_action(State,StateDesc,_), writelist(0,StateDesc),
    nl, write('has JUST BEEN COMPLETED.'), blank_lines(2),
    display_node_breakdown(State,0), try_again_message.

teaching_module(_,State,Report,4):- bad_consequence(State,Successor,BCD),
    node_action(State,StateDesc,_), node_action(Successor,SuccessorDesc,_),
    clear, user(Name),write(Name),
    write(', BEFORE  you can perform the action:'), nl,
    writelist(0,SuccessorDesc), nl, '
    write('YOU MUST FIRST  perform the action:'),
    nl,writelist(0,StateDesc),  nl,write('OTHERWISE '), nl,
    writelist(0,BCD),blank_lines(2), display_node_breakdown(State,0),
    try_again_message.

teaching_module(State,_,Report,5):-  bad_consequence(Prev,State,_), user(Name),
    clear, write(Name), write(', your PREVIOUS ACTION: '), nl,nl,
    node_action(Prev,PrevDesc,_),
    writelist(0,PrevDesc), nl,
    write('hasn"t been completed yet.  You SHOULD '),
    nl, write('HAVE REALIZED THIS by the LAST REPORT you received: '),
    nl,nl,
    writelist(0,Report), blank_lines(2),display_node_breakdown(Prev,0),
    try_again_message.

display_node_breakdown(State,No):- node_action(State,StateDesc,Type),
    Type=nex,
    write('The following action:  '), nl,
    writelist(0,StateDesc),nl,
    write('is a basic level action which doesn"t simplify ').nl,
    write('into subactions for further LEARNING.'),blank_lines(5),
    write('(Enter a "c." to continue.)'), read(C),clear,!.

display_node_breakdown(State,No):- node_action(State,StateDesc,Type),
    Type=tp, tp_file_type(State,Taskname,info), continue,
    consult(Taskname),clear,
    execute(Taskname,tp), !.
```

```
display_node_breakdown(State,No):- node_action(State,StateDesc,Type),
    Type=tp, asserta(order_steps(_)),
    tp_file_type(State,Taskname,_), continue, consult(Taskname), clear, nl,
    write('  ***  NEW SUB ACTION BREAKDOWN  ***  '), nl, nl,
    write('Below is a breakdown of the specific sub-actions,'),
    write(' which make up the action: '),
    nl, writelist(0,StateDesc), nl,
    operation_step(Step), save_states(Step), fail, !.

save_states(Step):- retract(order_steps(Operation_Steps)),
    abolish(order_steps,1),
    append(Step,Operation_Steps,NOp_Steps), asserta(order_steps(NOp_Steps)),
    !.

display_node_breakdown(State,No):- node_action(State,StateDesc,Type),
    Type=tp, retract(order_steps(Op_Steps)),
    length(Op_Steps,Len), write_no_list(Op_Steps,1,Len), continue, !.

display_node_breakdown(State,0):- nl,asserta(xstates([])),
    write('(Enter a "c." to continue.)'), read(C),clear,
    write('STUDY AND LEARN the following information to help you next time'),
    nl, write('you have a similar situation.  '), nl,
    write('Below is a breakdown of the specific sub-actions,'),
    write(' which make up the action: '),
    nl, node_action(State,StateDesc,Type), writelist(0,StateDesc),nl,
    display_node(State,5,1),!.

display_node_breakdown(State,1):- asserta(xstates([])),
    write('  ***  NEW SUB ACTION BREAKDOWN  ***  '), nl,
    write('Below is a breakdown of the specific sub-actions, '),
    write('which make up the action: '),
    nl, node_action(State,StateDesc,Type), writelist(0,StateDesc),nl,
    display_node(State,5,1),!.

display_node_breakdown(State,2):- asserta(xstates([])),
    write('  ***  REMINDER OF PREVIOUS ACTION BREAKDOWN  ***  '), nl,
    write('Below is a breakdown of the specific sub-actions, '),
    write('which make up the action: '),
    nl, node_action(State,StateDesc,Type), writelist(0,StateDesc),nl,
    display_node(State,5,2),!.
```

```
display_node(State.Index._):- node_action(Xstate.Xstatedesc._).
    compare_(Xstate.State.directlybelow).
    save_xstates( Xstate.Xstatedesc ).fail.!.
display_node(State.Index._):- get_xstates(Subactions).
    mapsecond(Subactions.ListTaskDesc).
    length(ListTaskDesc.Len).
    write_no_list(ListTaskDesc,1,Len), fail, !.
display_node(State._.1):- furtherbreakdownmess(State).
display_node(State._._):- retract(xstates(X)),!.

get_xstates(Subactions):- xstates(Subactions), !.

save_xstates(XstatePair):- retract(xstates(X)),
    append(XstatePair.X.NXstates).
    asserta(xstates(NXstates)).!.

choose_xstate(N.XstatePair):- xstates(XS), select(XS.N.XstatePair),clear.!.

furtherbreakdownmess(State):- blank_lines(1),
    writelist(0,['Do you want more breakdowns of these sub-actions?',
    ' INDICATE WHICH SUB-ACTIONS USING FORMAT "[#,#]."',
    ' otherwise ENTER "c." to continue.']),
    read(Choice),((Choice = c,!,fail);
    (!.member(Z.Choice),choose_xstate(Z.XstatePair).
    first(XstatePair,XstateNo),
    display_node_breakdown(State,2).nl.
    display_node_breakdown(XstateNo.1).
    nl,write('Enter "c." to continue: ').read(C).fail)).

try_again_message:- clear. blank_lines(5). user_name). write(Name).
    write(', now that you understand why your last choice of actions').nl.
    write('wasn"t the best choice. I"ll repeat the last '),
    write('report you received').
    nl. write('and the same choices. CHOOSE THE BEST ACTION. GOOD LUCK!').
    nl, write('(Enter a "c." to continue.)').read(C). clear.
```

VALIDATE INPUT

```
validate_inp(Inp,Max,Inp):- integer(Inp), Inp >= 1, Inp =< Max, !.
validate_inp(Inp,Max,NewChoice):- blank_lines(2), user(Name), write(Name),
    write(', your input was incorrect, please reenter your choice:'),
    read(Choice), validate_inp(Choice,Max,NewChoice), !.

validate(Choice,Choice):- choice(Choice).
validate(Choice,Inp):- nl, write('PLEASE enter a "yes." or "no.": '),
    read(Choice1), validate(Choice1,Inp),!.

choice(yes).
choice(no).
```

USER INTERFACE ROUTINES

DISPLAY USER RUNNING SCORE

```
interface(score,Pts,Total,Per):- clear, user(Name),
    pr_list_on_line(['                    '.Name.'''s Score  Level']),
    write('          '),
    pr_list_on_line(['                '.Pts.' of '.Total.'  '.Per.'%']),
    active_role(Role), pr_list_on_line([Name,', you are the "'.Role.'."']).
interface(score,Pts,Total,Per).
```

USER ACTION MENU GENERATOR

```
interface(menu,ActionList,UserAction):-
    length(ActionList,Len), mixup(ActionList,_,MxList,Len),
    menu(MxList,1,Len),
    N is Len - 3, read(Choice), validate_inp(Choice,N,Inp),
    ((req_aid(Inp,Len), last_report(_,Report),
    nl,write('YOUR LAST REPORT WAS: '), nl, writelist(0,Report),nl,nl,
    interface(menu,ActionList,UserAction));
    (select(MxList,Inp,UserAction))), !.

mixup([],X,X,0).
mixup(List,Oldlist,NList,Len):-random(Len,N), select(List,N,Item),
    append(Item,Oldlist,NewList), deleteone(Item,List,List1),
    Len1 is Len - 1,
    mixup(List1,NewList,NList,Len1).

list_menu(MxList,Index,Len):- select(MxList,Index,Item),
    node_action(Item,Action,_),
    write(Index), write('. '), writelist(0,Action), !, Index < Len.
    Index1 is Index - 1,
    list_menu(MxList,Index1,Len).

menu(MxList,Index,Len):- active_role(Role),
    write(Role), write(', indicate your most important,'), nl,
    write('              or next action to be completed.'),nl, nl,
    list_menu(MxList,Index,Len).
menu(MxList,Index,Len):- nl,user_aid(Len,1).
menu(_,_,_).

user_aid(Indx,I):- Indx1 is Indx + 1, !, aid(I,Aid,_),
    write(Indx1), write('. '), write(Aid), nl, I1 is I + 1,
    user_aid(Indx1,I1).

aid(1,'HELP',help).
aid(2,'QUIT',quit).

req_aid(UserChoice,Len):-  UserChoice > Len. N is UserChoice - Len.
    aid(N,_,Aidname).
    ((N=1, consult(Aidname), executeaid(Aidname),score_update(Aidname,0));
    (interface(grade), goodby)).
```

DIRECT USER TO COMPLETE TASK

```
interface(ncomp_action,Task,State):- nl, active_role(Role),
    write(Role), write(', you MUST now DIRECT THE ACTIONS to complete the '),
    nl, write('following TASK: '), node_action(State,StateDesc,_),
    writelist(0,StateDesc), nl.
```

ANNOUNCE CASUALTY

```
interface(announce_cas,HC,Casualty):- active_role(Role),
    write(Role), write(', you THEN receive the following report: '), nl,
    writelist(0,Casualty), nl, nl,sourcefile(_,HC),CasName,_),
    asserta(last_report(CasName,Casualty)).
```

TASK COMPLETION REPORT

```
interface(comp_action,Task,State):- nl,
    active_role(Role), write(Role),
    write(', you receive the following report: '), nl,
    write('The action: '), node_action(State,Action,NodeType),
    writelist(0,Action), write('has been reported completed.'), nl,nl,
    assign('The action: '.Action.' has been reported completed.',Report),
    check_last_report(Task),
    retract(last_report(Task,Rep)), asserta(last_report(Task,Report)),
    results_of_action(Task,State,NodeType), nl.
interface(comp_action,Task,State).


results_of_action(Task,State,NodeType):- random(2,N), ok_nok(N,Type),
    action_results(Task,State,Type,Results),
    length(Results,Len), random(Len,N1), select(Results,N1,Nresult),
    analyze_result(Task,State,Type,Nresult,NodeType).


ok_nok(2,ok).
ok_nok(1,nok).
```

```
analyze_result(Task,State,NewType,NewResult,NodeType):-
    retract(result(Task,State,OldType,_)),
    OldType = NewType,
    cleandatabase(Task,State,NodeType).

analyze_result(Task,State,NewType,NewResult,NodeType):-
    asserta(result(Task,State,NewType,NewResult)),
    active_role(Role), write(Role),
    write(', you THEN receive the following report: '), nl,
    writelist(0,NewResult), check_last_report(Task),
    retract(last_report(Task,Rep)),asserta(last_report(Task,NewResult)),
    cleandatabase(Task,State,NodeType).

cleandatabase(Task,State,tp).
cleandatabase(Task,State,_):- sourcefile(Task,State,File,_),
    retract(action_order(File,List)), retract(role(File,Role)),
    retract(start(File,_)), retract(previous_node(File,_)),
    cleansuccessor(File),cleansuccessorinfo(File),
    cleanresults(File), cleansource(File).
cleandatabase(Task,State,_).

cleansuccessor(File):- retract(successor(File,_,_)), fail.
cleansuccessor(File).

cleansuccessorinfo(File):- retract(successor_rule_info(File,_,_)),fail.
cleansuccessorinfo(File).

cleanresults(File):- retract(results(File,_,_,_)), fail.
cleanresults(File).

cleansource(File):- retract(sourcefile(File,_,_,_)), fail.
cleansource(File).
```

USER PERFORMANCE REPORT

```
interface(grade):- user(Name).
    write('Enter a "c." to continue.'). read(C). clear.
    writelist(24,'FINAL GRADE OF PERFORMANCE' ).nl.
    pr_list_on_line(['Congratulations ',Name,' you"ve survived!' ]). nl.
    write('Remember there"s always room for improvement in Damage Control').
    nl. write('Casualty Training.  Your life, your shipmate"s. and your '),
    nl. write('ship"s survival may someday depend on your KNOWLEDGE as a '),
    nl. write('MAN IN CHARGE of a casualty.  The following is a report of '),
    nl. write('your performance: '). nl, nl.
    write('Casualty    Task                Points   Total    Percent Grade').
    nl. overall_score(Pts,Total,Percent),
    write('Overall Score                   '),
    pr_list_on_line(['Pts.'      ',Total,'       ',Percent]),
    writelist(0,['Enter a "c." to continue.']), read(C),
    task_score(Task1,Pts1,Total1,Percent1). nl. describe_task(Task1,TaskDesc).
    writelist(0,TaskDesc),
    pr_list_on_line(['Pts = ',Pts1,'  Total = ',Total1,'  Percent Grade = ']).
    write(Percent1). fail.
```

FOLLOW ON TRAINING OPPORTUNITIES

```
interface(grade):- nl. write('Enter a "c." to continue.'). read(C). clear,
    writelist(0,['FOLLOW ON TRAINING OPPORTUNITIES']),
    write('For further training on the types of damage control actions or '), nl,
    write('equipment operations you just directed : '), nl,
    write('Enter a 1. To Quit'), nl,
    write('      2. Previous Fire Casualty Operations Training'). nl,
    read(Choice). validate_inp(Choice,2,Inp), ((Inp=1, goodby):
    (sort_task_list(TrainTasks),
    cleanentiredatabase. asserta(old_performance(TrainTasks)).
    initiate(5))), goodby.

cleanentiredatabase:- abolish(location,6), abolish(overall_score,3).
    abolish(task_score,4). abolish(result,4). abolish(action_order,2).
    abolish(successor,3).

sort_task_list(TrainTasks):- task_list(TaskScores).
    deleteone(mainfire,_,_,_,TaskScores,NewTaskScores).
    training_order(NewTaskScores,TrainTasks).

task_list(TaskScores):-
    bagof([Task,Pts,Total,Percent],
        task_score(Task,Pts,Total,Percent),TaskScores).
```

```
training_order( [] , [] ).
training_order( [X] , [X] ).
training_order(List, [L1,L2|N3List] ):- max_task(List,L1),
    deleteone(L1,List,N1List), min_task(N1List,L2),
    deleteone(L2,N1List,N2List),
    training_order(N2List,N3List).
```

File Name: execute_tp_info

INFORMATIONAL TEST PROCEDURE TASK EXECUTOR

Author: LT Steve Weingart, USN

```
execute_tp(Task,info):- asserta_dummy_location,
    tp_file_type(TPNo,Task,_), node_action(TPNo,TaskDesc,_),
    user_directions_info(TaskDesc),
    operation_step(Review), writelist(0,Review), continue, display_score,
    continue, remove_tp_info, retract(seed(Seed)), Seed1 is Seed + 57,
    asserta(seed(Seed1)).


user_directions_info(TaskDesc):- user(Name), clear, reference(Ref),
    write('          * * * KNOWLEDGE REVIEW  * * *   '),nl,nl,
    nl, write(Name), writelist(0,[', you will be shown a review of ',
    'important knowledge required in the performance of Task: ']), nl.
    writelist(0,TaskDesc), nl. nl,  writelist(0,[
    'STUDY AND LEARN THIS INFORMATION and for further guidance refer ',
    'to reference: '.Ref,
    'you will not be graded on this review.']), continue.


display_score:- overall_score(Pts,Total,Per),interface(score,Pts,Total,Per).
display_score.

remove_tp_info:- abolish(operation_step,1),
    abolish(task_name,1), abolish(reference,1).

asserta_dummy_location:-
    asserta(location(mainfire,'ADMIN OFFICE','Q','03','19','02')).

tp_file_type([1,1,1,1,5],chargefirehoses,info).
tp_file_type([1,2,1,2,1],choosedesmethod,info).
tp_file_type([1,2,5,1],debrief,info).
tp_file_type([1,1,1,1,2,2],determineroute,info).
tp_file_type([2,2,2,1],firstaid,info).
tp_file_type([1,2,2,3],firstoxygentest,info).
tp_file_type([1,1,1,1,2,1,2],idspace,info).
tp_file_type([2,2,1],removeinjured,info).
tp_file_type([1,2,2,4],secondoxygentest,info).
tp_file_type([1,1,2,3],setreflashwatch,info).
tp_file_type([1,1,2,1,2],activatehose,info).
```

File Name: execute_tp_order

TEST PROCEDURE ORDER DEPENDENT TASK EXECUTOR

Author: LT Steve Weingart, USN

```
execute_tp(Task,order):- tp_file_type(TPNo,Task,_), node_action(TPNo,TaskDesc,_),
    user_directions_order(TaskDesc),
    bagof(X,operation_step(X),Operation_sequence),
    length(Operation_sequence,Len), asserta(correct_seq(Operation_sequence,Len)),
    user_tp_test(Operation_sequence), remove_tp_order.

user_tp_test(Operation_sequence):-
    correct_seq(CorrectSeq,Len), asserta(testmenu(1,CorrectSeq)),
    mixup(Operation_sequence,_,MxTpList1,Len), asserta(testmenu(2,MxTpList1)),
    mixup(Operation_sequence,_,MxTpList2,Len), asserta(testmenu(3,MxTpList2)),
    menu_tp(1,Len).

menu_tp(Index,Len):- user(Name), task_name(Task),
    tp_file_type(TPNo,Task,_), node_action(TPNo,TaskDesc,_),
    clear, write(Name), write(', are the following steps the correct sequence to'),
    nl, write('perform the task: '),
    nl, writelist(0,TaskDesc), nl, nl, random(3,N), testmenu(N,MxTpList),
    write_no_list(MxTpList,Index,Len), nl,
    write('Enter a "yes." or "no.": '), read(Choice),
    validate(Choice,Inp), correct_seq(Operation_seq,_),
    ((Inp = no, analyze_no(Operation_seq,MxTpList,Index,Len));
    analyze_yes(Operation_seq,MxTpList)).

analyze_no(Operation_seq,MxTpList,Index,Len):- not(equal(Operation_seq,MxTpList)),
    menu_tp(Index,Len).
analyze_no(Operation_seq,MxTpList,_,_):- user_missed_correct_seq, task_name(Task),
    score_update(Task,0), continue.

analyze_yes(Operation_seq,MxTpList):- equal(Operation_seq,MxTpList),
    user_is_correct, task_name(Task), score_update(Task,1), continue.
analyze_yes(Operation_seq,MxTpList):- user_yes_is_incorrect,
    task_name(Task), score_update(Task,0),continue.

equal([X],[X]).
equal([X|List1],[X|List2]):- equal(List1,List2).

user_missed_correct_seq:- nl, nl, user(Name), write(Name),
    writelist(0,[', you are incorrect, the above sequence is correct!',
            'STUDY AND LEARN it before you continue.']), continue.
```

```
user_yes_is_incorrect:- user(Name), write(Name),
    writelist(0,[', you are incorrect. the above sequence',
    'is not correct! Below is the correct sequence. ',
    'STUDY AND LEARN it before you continue.' ]), nl, nl,
    correct_seq(Operation_seq,Len), write_no_list(Operation_seq,1,Len),
    continue.

user_is_correct:- nl, nl, user(Name), write(Name),
    writelist(0,[', your answer is correct.']), continue.

user_directions_order(TaskDesc):- user(Name), clear, reference(Ref),
    write('    * * * KNOWLEDGE TEST PROGRAM * * *   '),nl,nl,
    nl, write(Name), writelist(0,[', you will be shown various sequences of',
    'operational steps to perform the Task: ']), nl,
    writelist(0,TaskDesc), nl, nl, writelist(0,[
    'The reference for this task''s sequence of operational steps is the']),
    write(Ref),nl, nl, writelist(0,[
    'If the sequence is correct, you should enter a "yes." ',
    'otherwise a "no." if its incorrect. If your answer is incorrect,',
    'you''ll be shown the correct sequence of steps and given 0 points',
    'for this test. If your answer is correct you''ll be given 1 point',
    'for this test.',' ']), continue.


remove_tp_order:- abolish(task_name,1), abolish(operation_step,1),
    abolish(reference,1), abolish(testmenu,2), abolish(correct_seq,2).

tp_file_type([1,1,1,1,1,2,2],checkoba,order).
tp_file_type([1,2,3,1],determineamountwater,order).
tp_file_type([1,2,3,2],dewater,order).
tp_file_type([1,1,1,1,1,2,5],donfacep,order).
tp_file_type([1,1,1,1,4],estphonecoms,order).
tp_file_type([1,1,2,2],extinguishfire,order).
tp_file_type([1,1,1,1,1,2,4],instcan,order).
tp_file_type([1,1,1,3,3],manadjspaces,order).
tp_file_type([1,1,1,2,1],locatecirbrkr,order).
tp_file_type([1,1,1,1,1,2,3],placeonbody,order).
tp_file_type([1,2,1,2,2],removalpathest,order).
tp_file_type([1,2,1,1,3],sampleexspace,order).
tp_file_type([1,2,1,1,2],testexpequip,order).
tp_file_type([1,2,2,2],testoxygentester,order).
tp_file_type(1,help,tp).
tp_file_type(2,equipmentops,tp).
tp_file_type(3,dcactions,tp).
tp_file_type(5,prevtraining,tp).
```

File Name: mainfire

MAIN FIRE CASUALTY SCENARIO SOURCE FILE

Author: LT Steve Weingart, USN

```
successor(mainfire,[0,0,0,0],[1,1,1,1]):- initiate_task(mainfire),
    abolish(initiate_task,1).

successor(mainfire,[0,0,0,0],[1,1,1,1]).

successor(mainfire,State,Successor):-
    successor_rule_info(mainfire,State,Successor),
    not(result(mainfire,State,nok,_)).

successor_rule_info(mainfire,[1,1,1,1],[1,1,1,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,1,1,2],[1,1,1,3]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,1,1,3],[1,1,2,1]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,1,2,1],[1,1,2,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,1,2,2],[1,1,2,3]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,1,2,3],[1,1,2,4]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,1,2,4],[1,2,1,1]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,1,1],[1,2,1,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,1,2],[1,2,2,1]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,2,1],[1,2,2,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,2,2],[1,2,2,3]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,2,3],[1,2,2,4]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,2,4],[1,2,3,1]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,3,1],[1,2,3,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,3,2],[1,2,4,1]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,4,1],[1,2,4,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,4,2],[1,2,5,1]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,5,1],[1,2,5,2]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,5,2],[1,2,5,3]):- handlecasualty_code(0).
successor_rule_info(mainfire,[1,2,5,3],complete):- handlecasualty_code(0).
successor_rule_info(mainfire,_,[1,1,3,2]):- handlecasualty_code(1).


role(mainfire,'FIRE TEAM LEADER').
```

```
initiate_task(mainfire):-
 random(5,Level), random(100,Frame), random(6,Ctr),
 random(6,Type), space(Type,Desc,Sym),
 asserta(location(mainfire,Desc,Sym,Level,Frame,Ctr)),
 nl, write('DING    DING    DING    DING    DING'),
 nl, write('   FIRE    FIRE    FIRE    FIRE    '), nl,
 pr_list_on_line(
 ['FIRE IN ',Desc,', COMPARTMENT ',Level,'-',Frame,'-',Ctr,'-',Sym]),
 nl, write('AWAY THE REPAIR PARY FIRE TEAM AWAY!'), nl,
 asserta(result(mainfire,[0,0,0,0],ok,
 ['The ship has been informed of the location of the fire.'])),
 asserta(last_report(mainfire,
 ['The sh p has been informed of the location of the fire.'])).
 blank_l    s(3), display_role.

display     :-  user(Name), role(mainfire,Role),
    pr_l    on_line([Name,', you are now the "',Role,'."']), nl, nl.


space(1,'"A" DIVISION BERTHING','L').
space(2,'"E" DIVISION BERTHING','L').
space(3,'SUPPLY STOREROOM','A').
space(4,'ENGINEERING STOREROOM','A').
space(5,'PERSONNEL OFFICE','Q').
space(6,'ADMIN OFFICE','Q').



sourcefile(mainfire,[1,1,1,1],sceneequipready,ex).
sourcefile(mainfire,[1,1,1,2],deenergize,ex).
sourcefile(mainfire,[1,1,1,3],fireboundaries,ex).
sourcefile(mainfire,[1,1,2,1],approachfire,ex).
sourcefile(mainfire,[1,1,2,2],extinguishfire,tp).
sourcefile(mainfire,[1,1,2,3],setreflashwatch,tp).
sourcefile(mainfire,[1,1,2,4],verifyfireout,ex).
sourcefile(mainfire,[1,2,1,1],explosivegastest,ex).
sourcefile(mainfire,[1,2,1,2],desmoke,ex).
sourcefile(mainfire,[1,2,2,2],testoxygentester,tp).
sourcefile(mainfire,[1,2,2,3],firstoxygentest,tp).
sourcefile(mainfire,[1,2,2,4],secondoxygentest,tp).
sourcefile(mainfire,[1,2,3,1],determineamountwater,tp).
sourcefile(mainfire,[1,2,3,2],dewater,tp).
sourcefile(mainfire,[1,2,4,2],storeequip,ex).
sourcefile(mainfire,[1,2,5,1],debrief,tp).
sourcefile(mainfire,[1,2,5,3],securereflashwatch,ex).

start(mainfire,[0,0,0,0]).
```

File Name: equipmentops

MODE OF OPERATION:
INDIVIDUAL EQUIPMENT OPERATIONS AND
BASIC DAMAGE CONTROL ACTIONS

Author: LT Steve Weingart, USN

```
execute_tp(equipmentops,tp):- user_directions_equip_ops,
    bagof([TPnodeNo,TPdesc],node_action(TPnodeNo,TPdesc,tp),TestProc),
    mapsecond(TestProc,TestDescMenu), length(TestDescMenu,Len),
    write_no_list(TestDescMenu,1,Len), choose_tps(Choice),
    tp_test(Choice,TestProc), goodby.

user_directions_equip_ops:- user(Name), clear,
    write('          * * *  Equipment Operation Tests  * * *  '),
    nl,nl,nl, write(Name),
    writelist(0,[', you will be shown a list of equipment related operations',
    'and asked to enter which operations you want to review and be tested',
    'on.  A running score of your performance on your chosen equipment ',
    'operations will be displayed.  --  GOOD LUCK!  ']), continue.

choose_tps(Choice):- blank_lines(2), user(Name), write(Name),
    writelist(0,[', INDICATE WHICH OPERATIONS USING FORMAT "[#,#,#].".
    'ie. [1,3,4].']), read(Choice).

tp_test(Choice,TestProc):- member(Operation,Choice),
    tp_test1(TestProc,Operation), fail.
tp_test(_,_).

tp_test1(TestProc,Operation):-
    select(TestProc,Operation,TestItem), first(TestItem,TPNo),
    tp_file_type(TPNo,TPname,_), consult(TPname), execute(TPname,tp), !.
```

File Name: dcactions

MODE OF OPERATION:
INDIVIDUAL COMPLEX DAMAGE CONTROL ACTIONS

Author: LT Steve Weingart, USN

```
execute_tp(dcactions,tp):- asserta(random_yes_no(_,no)),assert_dummy_menu_items,
    user_directions_dcactions,no_files(NoFiles),
    bagof([TPnodeNo,TPdesc],node_action(TPnodeNo,TPdesc,ex),Tests),
    delete_same(NoFiles,Tests,_,TestProc),
    mapsecond(TestProc,TestDescMenu), length(TestDescMenu,Len),
    write_no_list(TestDescMenu,1,Len), choose_ex(Choice),
    ex_test(Choice,TestProc), goodby.

user_directions_dcactions:- user(Name), clear,
    write('          * * *  DAMAGE CONTROL ACTION TESTS  * * *  '),
    nl, nl, nl, write(Name),
    writelist(0,[', you will be shown a list of damage control related actions',
    'and asked to enter which actions you want to review and be tested',
    'on.  A running score of your performance on your chosen damage control ',
    'actions will be displayed.  --  GOOD LUCK!  ']), continue.

choose_ex(Choice):- blank_lines(2), user(Name), write(Name),
    writelist(0,[', INDICATE WHICH ACTIONS USING FORMAT "[#,#]."',
    'ie. [1,3,4].']), read(Choice).

ex_test(Choice,TestProc):- member(Action,Choice),
    select(TestProc,Action,TestItem), first(TestItem,TPNo),
    ex_file(TPNo,TPname), consult(TPname),node_action(TPNo,TaskDesc,_),
    task_instruction(TaskDesc),
    asserta(last_report(TPname,TaskDesc)),
    execute(TPname,ex),abolish(task_score,4), abolish(overall_score,3),
    nl, writelist(0,['THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:']),
    nl, writelist(0,TaskDesc), continue, fail.

ex_test(_,_).

task_instruction(TaskDesc):- user(Name), clear,
    write('     * * * DAMAGE CONTROL ACTION TEST * * *'), nl,nl,
    nl, write(Name), writelist(0,[', you may now direct the actions to ',
    'properly execute the action: ']), nl, writelist(0,TaskDesc),
    nl, nl,
    writelist(0,['Your score for this task will be displayed as you carry ',
    'out the actions. -- GOOD LUCK!']), continue.
```

```
assert_dummy_menu_items:- consult(mainfire). abolish(initiate_task,1).
    asserta(handlecasualty_code(0)). correct_action_order( 0.0.0.0 ,mainfire).
    asserta(location(mainfire,'PERSONNEL OFFICE','Q','2','89','03')).


no_files([[1],_],[1,1],_],[[1,2],_],[[1,1,1],_],[[1,1,2],_],[[1,2,1],_],
    [[1,2,2],_],[[1,2,3],_],[[1,2,4],_],[[1,2,5],_]]).

ex_file([2],injuredperson).
ex_file([3],brokenequip).
ex_file([1,1,2,1,1],openspacebound).
ex_file([2,2,2],medicalaid).
ex_file([1,1,1,3,3],closephysicalopening).
ex_file([1,1,1,1,1,1,1],getbattledress).
ex_file([1,1,1,1,1,2],getoba).
ex_file([1,1,1,1,1,3],getfireequip).
ex_file([1,1,1,1,1,3,1],putonhelmet).
ex_file([1,1,1,1,1,3,2],determinerequip).
ex_file([1,1,1,1,2,1],locatefire).
ex_file([2,2],careforinjury).
ex_file([1,1,1,1],sceneequipready).
ex_file([1,1,1,2],deenergize).
ex_file([1,1,1,3],fireboundaries).
ex_file([1,1,2,1],approachfire).
ex_file([1,1,2,4],verifyfireout).
ex_file([1,2,1,1],explosivegastest).
ex_file([1,2,1,2],desmoke).
ex_file([1,2,4,2],storeequip).
ex_file([1,2,5,3],securereflashwatch).
ex_file([1,1,1,1,1],getdressed).
ex_file([1,1,1,1,2],gettoscene).
ex_file([1,1,1,1,3],takecharge).
ex_file([1,1,1,1,3,3],relieveseniorperson).
```

File Name: prevtraining

MODE OF OPERATION:
PREVIOUS FIRE CASUALTY OPERATION TRAINING

Author: LT Steve Weingart, USN

```
execute_tp(prevtraining,tp):- user_directions_prev_training,standby,
    consult(dcactions), clear,
    old_performance(TrainTasks),
    member(Task,TrainTasks),.
    train(Task,TrainTasks,TaskDesc), nl,
    writelist(0,'THIS COMPLETES YOUR TEST OF THE DAMAGE CONTROL ACTION:'),
    nl, writelist(0,TaskDesc),
    continue,abolish(overall_score,3),abolish(task_score,4),fail.
execute_tp(prevtraining,tp).

train(Task,TrainTasks,TaskDesc):- first(Task,TaskName),
    describe_task(TaskName,TaskDesc), asserta(last_report(_,TaskDesc)),
    nl, (node_action(TPNo,TaskDesc,NodeType);true),
    ((atom(TaskName), assign(TaskName,NTaskName)):
    ((NodeType=tp,tp_file_type(TPNo,NTaskName,_));
    (ex_file(TPNo,NTaskName)))),
    consult(NTaskName), clear,
    user(Name),nl, write(Name),
    writelist(0,', your task will be:'),
    writelist(0,TaskDesc), nl, percent(Task,Percent),
    p_list_on_line(
    'Your previous performance grade was ',Percent,'for this task.'),
    !, execute(NTaskName,NodeType), abolish(last_report,2),
    !.

user_directions_prev_training:- user(Name), clear,
    write('          * * *  PREVIOUS TRAINING REVIEW  * * *  '), nl,
    nl, nl, write(Name),
    writelist(0,', you will be retested on the tasks which you were ',
    'just tested on.  I''ve taken the liberty of ordering the tasks',
    'with your best first, then your worst second, then your second',
    'best third, then your second worst fourth and so on. ',
    'I think you''ll enjoy this order of retraining the best.',
    '                GOOD LUCK!  '), continue, !.

percent(_,_,_,Percent,Percent).
```

LIST OF REFERENCES

1. Naval Sea Systems Command Report S9086-CN-STM-020, Naval Ship's Technical Manual, Chapter 079, Volume 2, Section 42, Damage Control-Practical Damage Control, December, 1979.

2. Reiser, B.J., Anderson, J.R., Farrell, R.G., Dynamic Student Modelling In An Intelligent Tutor For Lisp Programming, Advance Computer Tutoring Project, Carnegie-Mellon University, Pittsburgh, PA 15213.

3. Barr, A. and Feigenbaum, E.A., The Handbook of Artificial Intelligence, V.2, William Kaufmann, Inc., 1982.

4. Clancey, W.J., GUIDON, Stanford University, Palo Alto, California.

5. University of Pittsburgh Learning and Research and Development Center Report UPITT/LROC/ONR/APS-15, SPIRIT: An Evolutionary Designed Tutoring System, by A. Barzilay and H.E. Pople, Jr., July 1984.

6. Anderson, J.R., Boyle, C.F, and Yost, G., THE GEOMETRY TUTOR, Advanced Computer Tutoring Project, Carnegie-Mellon University, Pittsburgh PA 15213.

7. Rizza, Peter J., "Computer-Based Education (CBE): Tomorrow's Traditional System," Children in Contemporary Society, v. 14, Fall 1981.

8. Naval Sea Systems Command Report 0901-LP-930-0003, Naval Ships Technical Manual, Chapter 9930 Damage Control-Practical Damage Control, 1967.

9. Bissel, A.M., Oertel, E.J., and Livingston, D.J., Shipboard Damage Control, Naval Institute Press, 1976.

10. Schneider, G.M., and Bruell, S.C., Advanced Programming and Problem Solving with Pascal, John Wiley & Sons, Inc., 1981.

11. Hayes-Roth, F., Waterman, D.A., and Lenat, D.B., Building Expert Systems, Addison-Wesley Publishing Company, Inc., 1983.

12. Prolog-86 (tm) Version 1.12 Copyright 1983, 84. Micro-AI, Moraga, Ca.

13. Clocksin, W.F. and Mellish, C.S., Programming in Prolog, 2d ed., Beltz, Offsetdruck, Hemsbachm, 1984.

14. Winston, P.H. Artificial Intelligence, Addison-Wesley Publishing Company, Inc., 1984

BIBLIOGRAPHY

Naval Education and Training Command Rate Training Manual 10054-E1, Basic Military Requirements, 1981.

Naval Education and Training Command Rate Training Manual 465-08-00-82, Combustible Gas Indicators and Flame Safety Lamp, 1982.

Naval Education and Training Command Rate Training Manual 465-07-00-77, Desmoking and Emergency Ventilation Equipment Red Devil and Pneumatic Blower, 1977.

Naval Education and Training Command Rate Training Manual 465-10-00-77, Dewatering Equipment, 1977.

Naval Education and Training Command Rate Training Manual 465-02-00-83, Fire and Firefighting, 1983.

Naval Education and Training Command Rate Training Manual 465-03-00-79, Firehose and Fitting, 1979.

Naval Education and Training Command Rate Training Manual 10520-G, Fireman, 1982.

Naval Education and Training Command Rate Training Manual 10574, Hull Maintenance Tech 1&C, 1972.

Naval Education and Training Command Rate Training Manual 465-04-00-77, Nozzles and Applicators, 1977.

Naval Education and Training Command Rate Training Manual 0501-LP-221-1925, Personnel Qualification Standard For Damage Control, February, 1980.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5000	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
4.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, California 93943-5000	1
5.	Associate Professor N.C. Rowe Code 52Rp Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
6.	Associate Professor M.J. Zyda Code 52Zk Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
7.	Commanding Officer Naval Research Laboratory Washington, DC 20375	1
8.	Commanding Officer Naval Damage Control Training Center Philadelphia, PA 19112	1
9.	Chief of Naval Education and Training Naval Air Station Pensacola Pensacola, Florida 32508	1
10.	Professor R. McGhee Code 52Mz Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	1
11.	Professor David L. Smith Code 69Sn Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93943-5000	1
12.	LT Stephen G. Weingart, USN 3669 Halifax Ave No. Robbinsdale, Minnesota 55422	2

END

12-86

DTIC